THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR DONALD LEE

TITLE OF THESIS GRAPHICS INPUT AND ANALYSIS FOR COMPUTER-ASSISTED INSTRUCTION

DEGREE FOR WHICH THESIS WAS PRESENTED MASTER OF EDUCATION

YEAR THIS DEGREE GRANTED FALL 1983

THE UNIVERSITY OF ALBERTA

GRAPHICS INPUT AND ANALYSIS FOR COMPUTER-ASSISTED INSTRUCTION

by

DONALD LEE

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF EDUCATION

DEPARTMENT OF SECONDARY EDUCATION

EDMONTON, ALBERTA

FALL 1983

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled GRAPHICS INPUT AND ANALYSIS FOR COMPUTER-ASSISTED INSTRUCTION submitted by DONALD LEE in partial fulfilment of the requirements for the degree of MASTER OF EDUCATION.

Abstract

Current computer-assisted instructional (CAI) systems provide only a limited means for student entry of responses, usually keyboard or keyboard augmented by lightpen. Generally, such systems can only accommodate multiple choice, short answer or a single Cartesian coordinate location pointed to by a lightpen. A new tool for extending the input mechanism of CAI systems to handle graphics input and provide an analysis with respect to predefined target graphics images is investigated. The mechanism functions by using a decision matrix whose operational state is decided by the type, number and occurrence of graphics data primitives found in the student input and target images. Each cell of the matrix indicates which analysis levels of a hierarchy of analysis routines are thus suitable for analyzing the data structures. In this sense, the hierarchy is constrained by the target image(s). Preliminary results from an implemented subset of this mechanism, known as Graphics Matching Algorithm using Target Constrained Hierarchy (GMATCH) are encouraging.

The reasonable man adapts himself to the world: the unreasonable one persists in trying to adapt the world to himself. Therefore all progress depends on the unreasonable man.

George Bernard Shaw *Man and Superman*

Most importantly, man can foresee .... His tools, whatever their primary practical function, are necessarily also pedagogical instruments. They are then part of the stuff out of which man fashions his imaginative reconstruction of the world.

Joseph Weizenbaum *Computer Power and Human Reason*

Acknowledgements

I am deeply indebted to Dr. T.E. Kieren for his insight, guidance and never-ending patience while serving as my supervisor. I am especially grateful to Dr. S.M. Hunka for his encouragement, valued suggestions and for the crucial assistance, in the form of hardware support and general comments, provided by the staff and graduate students at the Division of Educational Research Services (in particular, the contributions of J. Hunka, A. Davis, N. McGinnis, J. Nesbit and S. Bushey are much appreciated). Finally, I also wish to thank Dr. A.T. Olson for his comments, interest and commitment.

Table of Contents

List of Tables

List of Figures

I. INTRODUCTION

A. The Problem

A new era in computer-assisted instruction (CAI) has been anticipated with the advent of low-cost hardware, most notably microcomputers with high-resolution color graphics capability. To date, however, expectations of a renaissance have not yet materialized. Both critics and proponents of CAI consistently allude to a dearth of good courseware as a primary hindrance to the acceptance and growth of CAI. By implication one source of this problem may be attributable to a lack of good authoring software. Many of the current CAI authoring languages and systems owe their origins to software developed at least two decades ago. Consequently, much courseware design still exemplifies the limited and often simplistic instructional strategies that were in vogue at that time.

A current review of some contemporary authoring systems and languages (WICAT, 1982) reveals that the ability to analyze student response (input) is restricted to an analysis of keyboard input of characters or sentences (string-matching) or to the determination of a screen area or location on the CRT by some pointing instrument such as a lightpen or a touch-sensitive screen. If one posits the notion that the ideal CAI system should embody the best abilities of a human teacher to recognize, analyze and classify student responses, then clearly the current

authoring languages are insufficient. A human teacher can accept, judge or validate free-form visual or graphics-oriented responses, whereas current implementations of CAI languages provide a forced choice from a short list or, at best, a textual response via a keyboard. Even when such systems are capable of generating high-resolution graphics, their ability to analyze student input is usually restricted to a determination of a Cartesian coordinate location or an ASCII character at that location (Romaniuk, 1970; Voyce, 1979)[1] As an example, current CAI practice would request a student to select one of four curves drawn on a CRT display as the answer to the question, "What is the graph of $y = \cos^2 x$?" This task, however, is not identical to that of calculating and drawing the curve of $y = \cos^2 x$. In this instance CAI would be limiting the representation of a mathematical idea by a student even though such representations might be critical to the learning of the idea.

The general stress towards the use of symbolic information is not necessarily the result of a lack of hardware sophistication, but rather appears to be the

---

[1] It should be noted that some CAI systems do have the potential to allow student creation of graphics images. Stan Smith, for example, has created some interactive chemistry simulations on Plato (Sugarman, 1978) which permit students to build a distillation apparatus from prespecified component images. However, the cost in terms of authoring time and programming skill to permit this facility generally precludes its availability on most CAI systems. One exception is the TIGSI system (Nygard and Ranganathan, 1983), which has the capability to accept student input data and generate 2-D displays interactively.

product of our mechanism of formal education which has long depended on verbal mechanisms (Bork, 1981). A similar position was taken by Huggins (1974) in a scathing address concerning the lack of use of graphics in education.

> Higher education certainly becomes increasingly symbolic....Once we're able to give a symbolic description of something, we consider a pictorial representation a step backwards. (p. 35)
>
> Instead of displaying iconic skills, we rely on symbols and assume the student knows what we mean. But after all, the symbol is just a mark or a sound that has no meaning in itself. And the student drifts around wondering what all this means--manipulating symbols formally like crazy because he knows the syntactical rules, but doesn't know the meaning. (p. 36)

These remarks, while hyperbolic and simplistic, reflect a wide-spread perceived ill about our educational system that cannot be readily dismissed. While extensive use of symbolic representation may be characteristic of highly-industrialized cultures, the disparity of emphasis on symbolic representation without consideration for alternative modes of presentation imposes particular problems for students who are less capable of assimilating purely verbal information. Even for students who are verbally competent, alternate graphical modes can enhance and strengthen conceptual understanding. However, as hypothesized by Bork (1975, pp. 287-290), man's iconic skills have greatly diminished as his abilities to communicate verbally have increased.[2]

---

[2]The acceptance of LOGO as a means to bridge the gap between the constructs of a symbolic language and its graphical equivalent would seem to be evidence of this.

It is, therefore, not surprising that the most modern CAI authoring systems and languages provide capable semantic-type answer judging facilities while neglecting the facility for graphics input. Fields and Paris (1981) allude to this need when they state:

> training in some subject domains, like architecture, requires drawing as input to the CBI system. There is a special problem in making sense of drawing input, which looks to the computer like a series of disconnected "points" or dots rather than as the revolutionary building design, which the student sees. (p. 70)

Having identified a need for graphics input to augment the current capabilities of CAI, one should determine the fundamental characteristics of a tool to provide this ability. Naturally, there must also be a means for comparing or judging the merit of a student response, an analysis component. That is, given a situation in a CAI course in which graphics input is appropriate, one major task of the tool is to compare the student input image against a target, an image or a description of a process previously created by an author. This study provides a formal specification of such a tool, its design and mechanisms of operation, its implementation, and its evaluation.

B. Informal Specifications for Graphics Input and Analysis

A new CAI tool, designed to accept, analyze, and classify student responses in the form of graphics images, is proposed as an essential, evolutionary step in the development of more powerful and better CAI software and

hence, courseware. Four major components can be identified:

1. In order to be useful, this tool should provide for the creation of sets of graphics images to be used as targets in a easy fashion -- an author interface.
2. Similarly, the student who interacts with the CAI system by submitting graphics input to a question should require minimal training in order to use its features -- a viable student interface.
3. The creation and analysis of graphics images requires a means for expressing that image in a manner that can be manipulated by a computer, in other words, a data structure.
4. An algorithm to analyze the student input versus the author-specified target must be developed. The results of the comparison must be translatable to some figure of merit or value that is meaningful to both author and student.

A Graphics Matching Algorithm using Target Constrained Hierarchy (GMATCH) procedure has been developed as an attempt to meet these requirements.

C. Limitations and Delimitations

GMATCH as presented here has been developed and tested at a level constrained by the following limitations.

1. Hardware availability and constraints prevented full-scale implementation and testing of the GMATCH module. (See Chapter Four for a complete description.)

2. The GMATCH module was implemented independently of any CAI environment.
3. In situe testing was not performed due to item 2 above.

The following delimitations are noted:

1. The study is restricted to the input and analysis of two-dimensional figures only, although consideration is given to three-dimensional development in Chapter Five.
2. The study does not consider any formal specifications of a graphics presentation language per se.
3. Evaluation of GMATCH is restricted to CPU response times.

D. Organization

Chapter Two provides an introduction to the pattern recognition research by consideration of some fundamental concepts. This leads into an overview of some pertinent studies in both data structuring and classification. GMATCH is formally described in Chapter Three, starting with an examination of the overall CAI environment and is followed by the formal specifications for the data structures, a description of the author and user interfaces, the elements of the GMATCH hierarchy, and finally the internal operations of the analysis within each level of the hierarchy. The results of some initial tests using an implemented subset of GMATCH on an existing CAI facility is documented and discussed in Chapter Four. Finally, Chapter 5 deliberates the import of GMATCH with respect to pedagogical design, its

ability to handle dynamic and three-dimensional input, and its future growth.

A glossary is included in Appendix A, Appendix B describes the RGB color space, and Appendix C illustrates some of the target data sets used in the preliminary testing of a GMATCH subset.

II. OVERVIEW OF RELEVANT RESEARCH AND LITERATURE

A. Limitation of scope

With the sole exception of Fields and Paris (1981), no other literature on the topic of **analysis of graphics input** within the CAI domain has been discovered. Fields and Paris' prognostications are essentially summarized in the following:

> It is unlikely that image or sketch understanding systems will be useful for CBI system designers within the next decade, primarily because of the low level of research and development (R & D) investment devoted to finding new algorithms. (p. 84)

Their skepticism notwithstanding, this study demonstrates that the current hardware and software technology is capable of rendering some useful image recognition (matching) procedures for CAI.

The majority of the relevant research originates from that branch of artificial intelligence associated with pattern recognition or image analysis. Two general classifications of techniques exist: the statistical method and the syntactic method. Due to the extensiveness and diversity of research found in both approaches, this review attends only to those algorithms or procedures that are either directly related or are of technical and theoretical significance to this study.

B. Some fundamental concepts

Given that there exists some picture function G (which may be multi-valued), such that G = (x,f(x)) where x, f(x) ϵ R, then it is assumed that there exists a mapping of G from RxR to IxI (a digitization). That is, G, which consists of an infinite number of points in Real space, has a mapping in Integer space, H, which consists of a finite number of points such that the fundamental features of G are not lost under the transformation (Figure 1). In order to represent this digitized picture or image by means of a computer there must be a means of representing H in a form that is internally manageable by a machine. This form is known as a data structure. The data structure also determines the bit configuration of the image within the physical entity known as the video buffer or video RAM. This bit configuration provides the information necessary to control the picture elements (pixels) of the monitor or CRT device (Figure 2). Although it is not completely accurate, it is often useful to imagine the video RAM as an n x n matrix of cells. Each cell contains either a zero or a one (off-state or on-state of a pixel, respectively). Hence, one can imagine a mapping directly from H onto video RAM with each bit of video RAM corresponding to a point in H. Thus, the state of video RAM would appear to be a digitized, binary image of H.

The data structure which directly controls the display of the image on a CRT may or may not be the same as the data

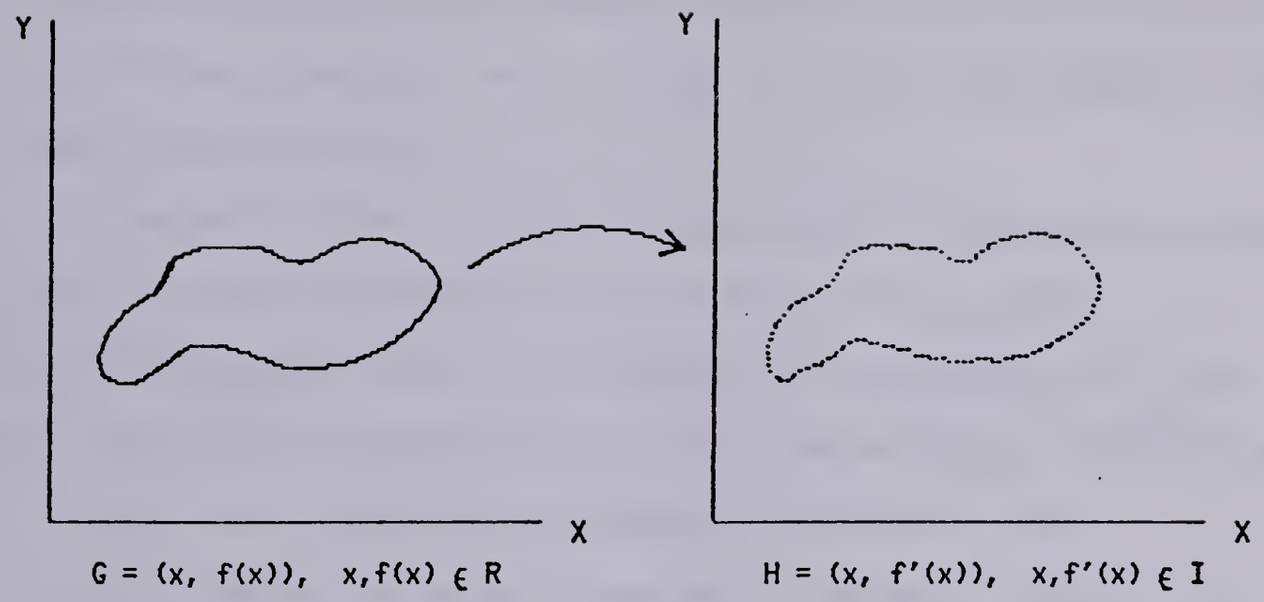


Figure 1 Mapping of G onto H

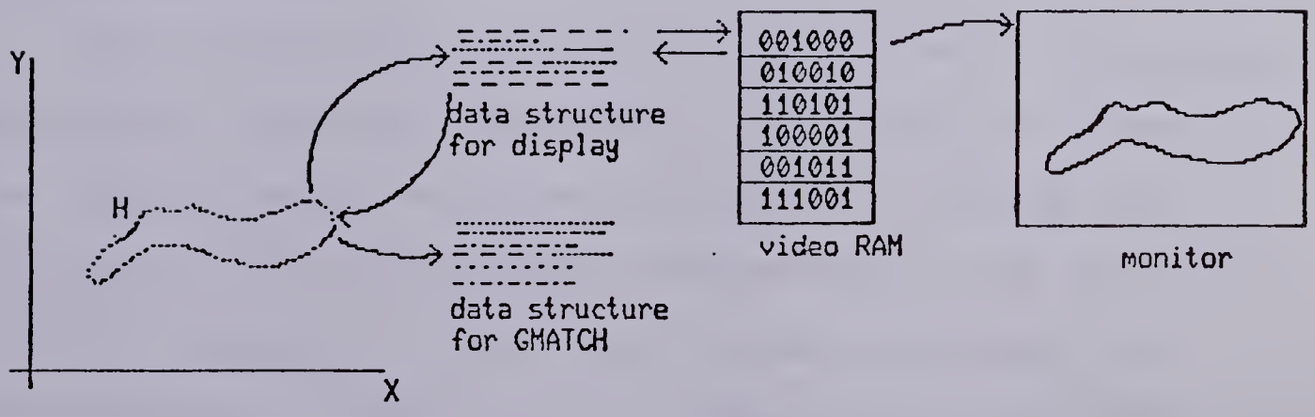


Figure 2 Mapping of H into Data Structure and CRT

structure for image matching analysis. In this study, a distinction is made between the data structures supporting the graphics display and those which are to serve as vehicles for analysis by GMATCH. Unless otherwise stated the term "data structure" will refer to that which supports the image matching task.

Assuming that an input image exists in RAM and a target image is also provided in matrix form, it is possible to directly compare these two images by performing a bit for bit check for correspondence. This methodology, which often involves image-shifting hardware to maximize the correspondence, is often known as correlation (due to the possibility of confusion, the term "correlation" will henceforth be used strictly in the statistical sense as a measure of correspondence). A close match is said to exist if most of the bits (or pixels) of the input correspond with those of the target. The target serves as a "template" against which judgement must be made (Figure 3).

C. Data structures

One of the major constituents of any model for pattern recognition, whether statistical or syntactic in nature, is the representation or the data structure. Although the structure chosen may be quite independent of the technique used for analysis, in practice, the data structure is selected so as to facilitate the analysis. An extremely simple approach, which is still used to great advantage

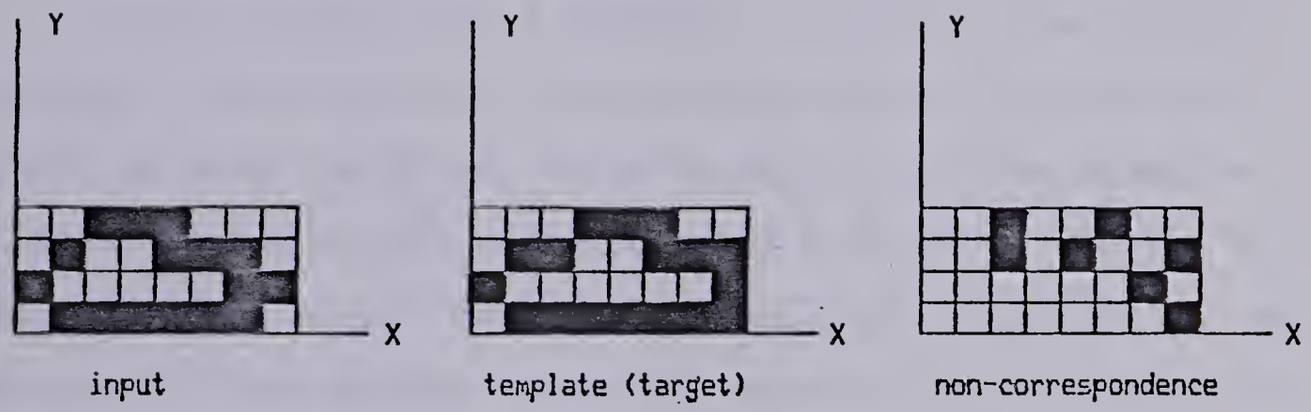


Figure 3 Comparison of Input and Target (Template)

(Edmonds, Schappo, & Scrivener, 1982), is to represent the image or object by means of a one-to-one mapping of the points defining the image with the cells of an n x n matrix. Thus, a black and white image willl have a binary matrix representation (white=1, black=0, or vice-versa) which is visually suggestive of the image itself. The primary disadvantage of this approach is the lack of conciseness of coding. A variant of this approach is to specify the cells of either the 0-state or the 1-state as a set of Cartesian coordinates:

$$f(p) = \{(x_1,y_1),(x_2,y_2),(x_3,y_3),\ldots,(x_n,y_n)\}$$

Another variation which, for images containing blocks of consecutive cells of the same state, is more compact is a

run-length encoding form:

$f(p) = \{k_1(x_1,y_1),k_2(x_2,y_2),k_3(x_3,y_3),\ldots,k_n(x_n,y_n)\}$,

where k_i is a constant denoting the number of cells of the same state which follow (in the same row) starting at position (x_i,y_i).

A more sophisticated approach devised by Blum (1967) yields a transformation known as the medial axis function (MAF) or Blum transform, which extracts from the object a representation highly reminiscent of a skeleton (Figure 4). The MAF of an object consists of the locus of points of the centers of the maximally empty circles and their respective radii such that the envelope of circles describes the boundary of the object. As a means for a concise expression of an object, the Blum transform suffers from a severe sensitivity to local distortions (see Pavlidis, 1968) of the object boundary as is evident in Figure 5.

A chain-coding technique used by Freeman (1974) employs a clockwise traversal procedure to cover the boundary of the object by moving from pixel to pixel using vectors. The resultant chain of vectors is coded as a numerical sequence of digits ranging from zero to seven. Since the figure is closed, the sequence is cyclical but the left-most vertex of the figure may be chosen as the origin to ensure uniqueness of representation (Figure 6).

Udupa and Murthy (1975) devised a coding scheme that is similar to chain-coding. However, they applied small matrix operators, typically 3x3 in size (this concept is detailed

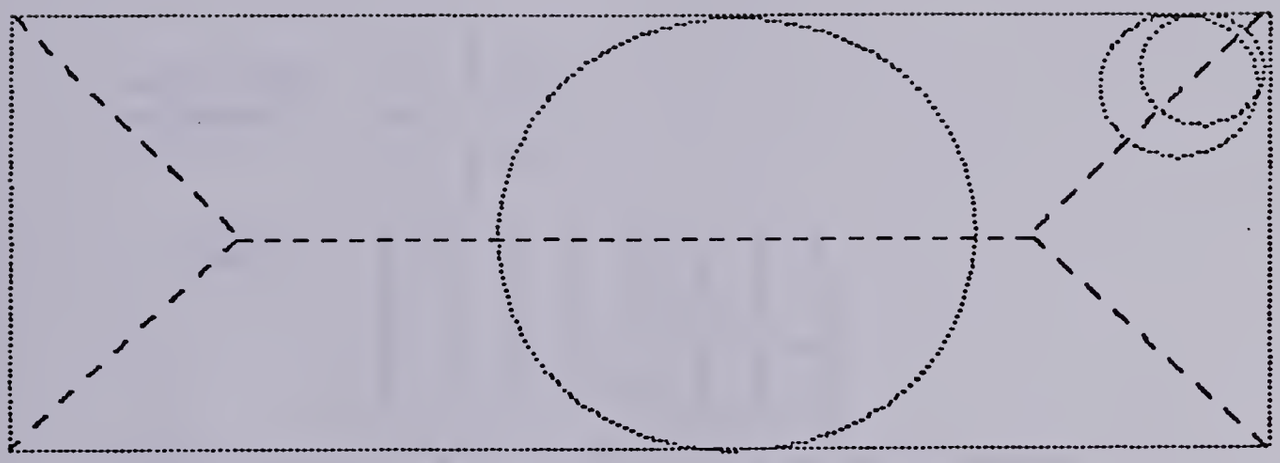


Figure 4 The MAF of a Rectangle along with some Maximal Circles

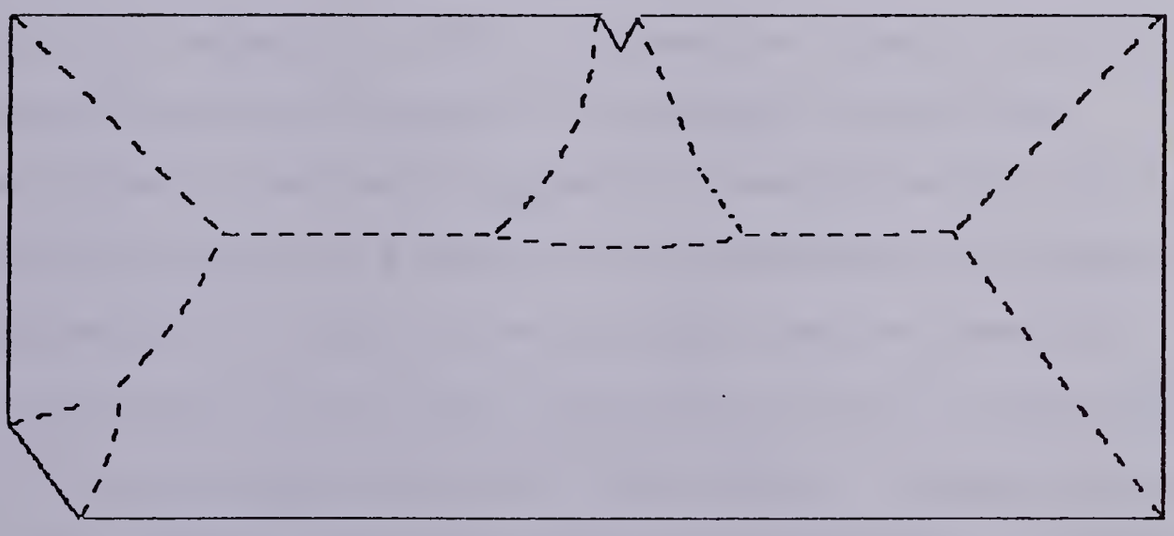


Figure 5 Generation of New Branches as a result of Boundary Distortions

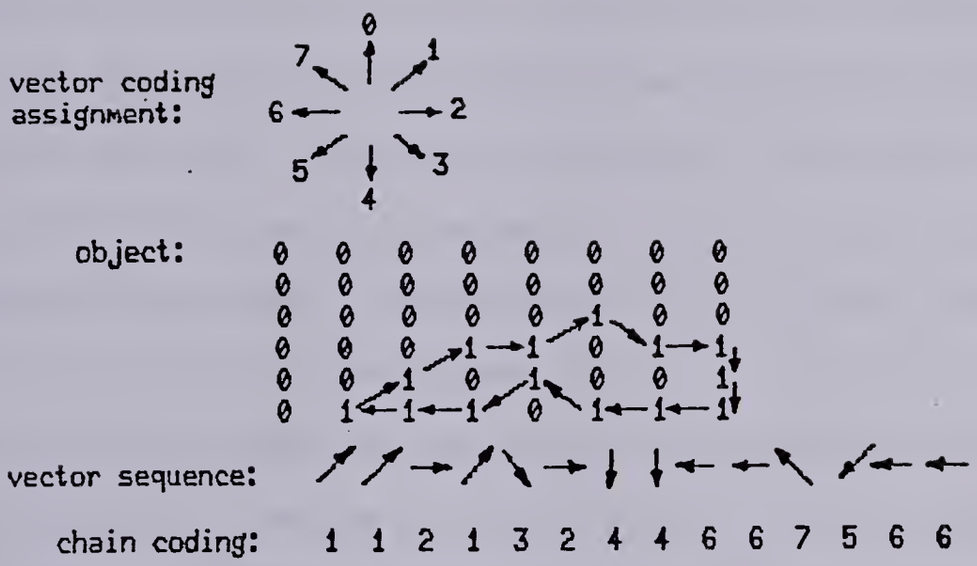


Figure 6 Chain-coding Technique

in Section D, Classification Techniques), to condition the original binary geometric pattern. This produces locations in the figure which are described as either turning points (T's) or endpoints (E's). A subscript notation is then applied to the T's and E's to produce a direction assignment. The resulting matrix consists mostly of 1's with a scattering of T's and E's at significant locations. A sequence of T's and E's which describes the general appearance of the figure can then be built up. This code is noted to be scale invariant and rotation variant only by a fixed constant applied to the first subscript of the T's and E's in the sequence.

In his study of handprinted characters, Nevins (1979) also used the concept of a chain of vectors as a preliminary step in representing a graphics character. Each chain of vectors is then analyzed by examining all groupings of three consecutive vectors for the occurrence of an inflection point. An inflection point separates the chain into convex arcs called links. Since the inflection point must occur at some point in the middle segment of the triad of vectors, two possible arbitrary assignments can be used. That is, the inflection point can be chosen to be at the start of the vector or at the end of the vector. Subsequent inflection points found in the chain will take alternating assignments as to their location. Hence, two possible codings can be used to describe the chain. One coding consists of the links and corresponding inflection points derived from the alternating sequence of beginning point, end point, beginning point, and so forth; while the other coding is derived from the end point first representation. Objects thus classified can be placed within a family description if three conditions are met:

1. equivalent number of sections for both objects,
2. corresponding sections have same number of chains, and
3. corresponding chains have same number of segments emanating from their beginning points, end points and in the number of links.

A more recent approach described by Nevins (1982) extracts primitive convex regions from a complex shape.

These regions are then represented as nodes of a tree with the branches between nodes corresponding to the common edges between regions. On the other hand, Grosky and Jain (1983) partition the binary n x n matrix containing the image into quadrants. Starting at the upper left quadrant and working clockwise, each quadrant which is uniform (all 0-state or all 1-state) is represented as a leaf in the resultant tree diagram. A non-uniform quadrant is represented as an internal node in the tree. Each non-uniform quadrant is, in turn, successively divided into its quadrants until the tree structure results in only leaves. Figure 7 shows an 8 x 8 example of this process.

For small images the quadtree representation yields a structure that is relatively compact but as the size of the matrix increases, say 1024 x 1024, the tree structure becomes rather immense, particularly for large irregular shapes with thin boundaries. Analysis of such large structures becomes both complex and time-consuming.

The data structures examined thus far, with the exception of the MAF, do not embody any operational definitions. Such a structure has been proposed by Agui, Nakajima, and Arai (1982). Their algebraic approach provides a means for both description and generation of binary images for display. Neglecting the formalisms of their algebra, any binary image is represented by the summation (exclusive OR) of terms in the polynomial

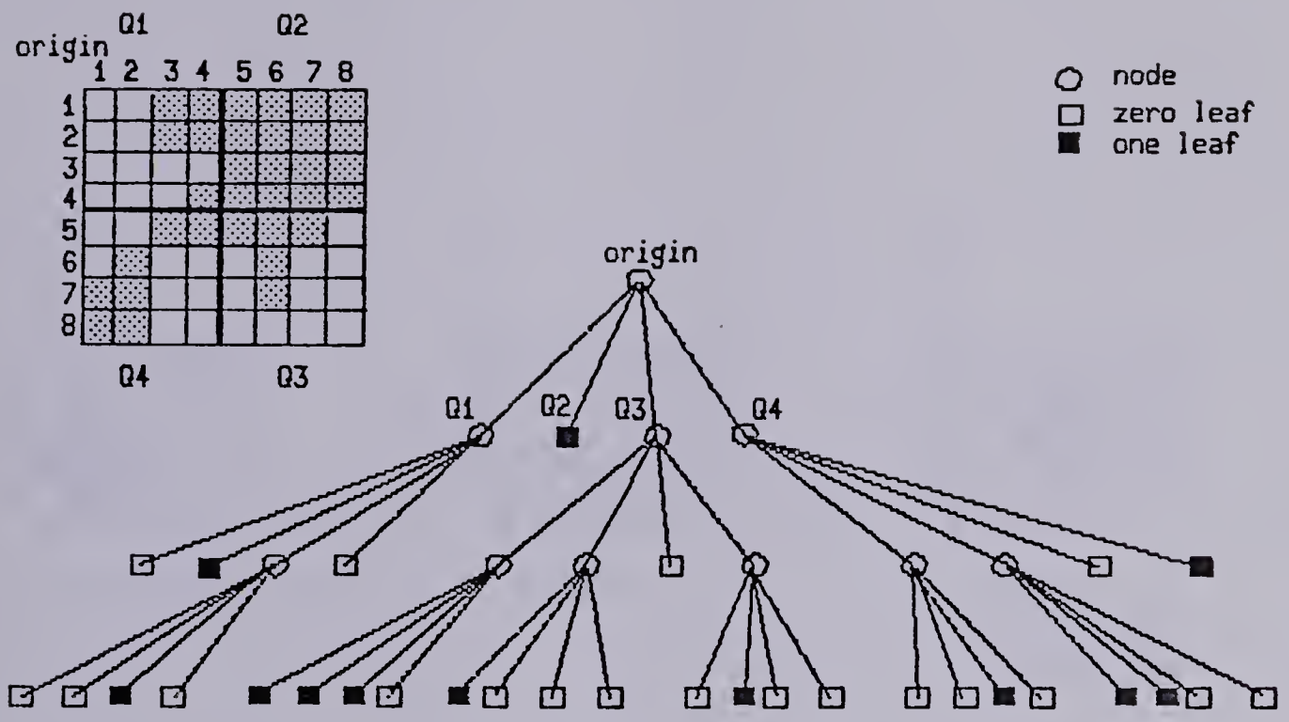


Figure 7 Quadtree Representation of a Binary Image

$$\sum_{(p,q)\in A} x^{P} y^{Q}$$

where p, q are the (x,y) coordinates of the image points. Hence, multiplication of one image by a single pixel is equivalent to an affine transformation of type translation (Figure 8). The algebra of Agui et al. specifies an image exactly but a large image (dimensions greater than one hundred pixels) would generate an excessive number of terms in the polynomial expansion, thereby impacting subsequent analysis.

The aforementioned data structures, while representative of the field of image and pattern analysis, are by no means exhaustive. Many variants of these structures exist and are more thoroughly treated in the

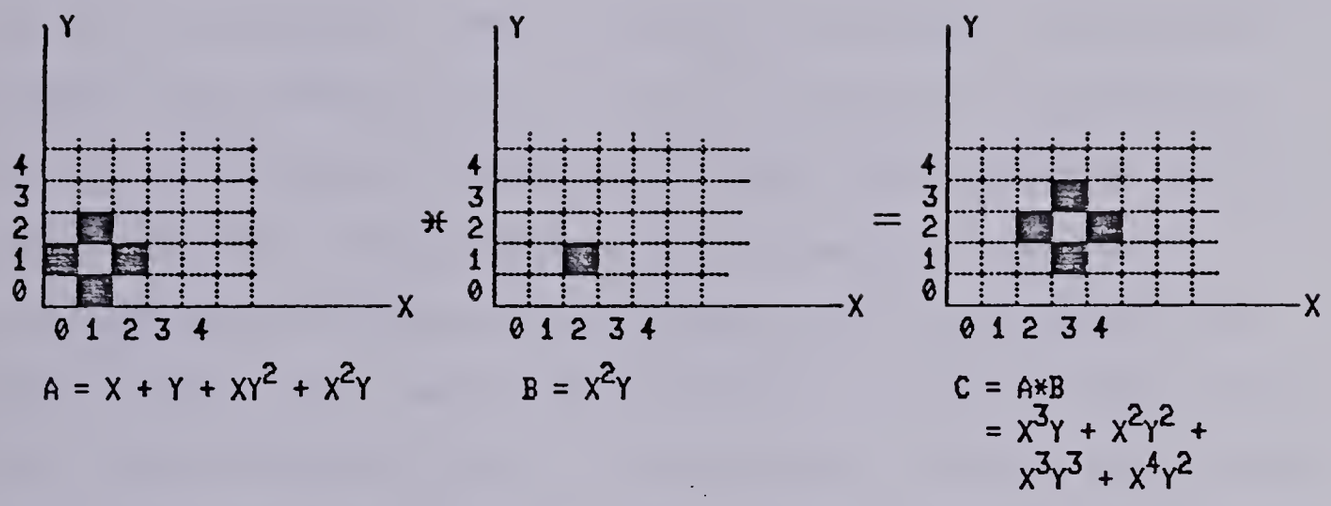


Figure 8 Multiplication of Two Images using a Polynomial Approach

books of Duda and Hart (1973), Hall (1979), and Fu (1982). A more formal treatment applied to the area of graphics programming languages (not pattern recognition) is found in an ACM distinguished dissertation by Mallgren (1982).

D. Classification Techniques

The other two constituents in a general model for pattern recognition are feature extraction and image classification which are considered here as a single entity. Feature extraction refers to the identification of key attributes or fundamental characteristics of an image. As an example, the features of an image may include the ratio of major axis length to minor axis length, the number of 90

degree angles, the angle between two adjacent edges, the coordinates of the inflection points, and so forth.

One of the earliest and highly successful attempts at feature extraction and classification was a program invented by Uhr and Vossler (1963). Five by five matrix operators whose cells consist of 0's, 1's or blanks are translated across and down the scaled, digitized, binary image of the input. At each cell location of the image the number of correspondences between the operator and the subimage are determined. A hit is registered when all the 1's and 0's of the operator match those of the subimage (Figure 9). At each hit location four characteristics are measured and coded as a binary number. As each operator in turn scans the image, a sequential list of these characteristics is built and hence, a sequence of digits results. The differences between the input pattern's sequence and other known patterns' sequences (maintained as lists in memory) is operated on by lists of three types of weighting factors (called amplifiers) to produce a difference score. The input is identified to be the pattern with the lowest difference score. After identification the amplifiers are tuned such that those characteristics which ultimately contribute more to the recognition of the input pattern are weighted more while the other characteristics are downgraded.

The construction of the matrix operators may be predetermined by the experimenter, randomly generated by the program, randomly extracted from the input matrix, or

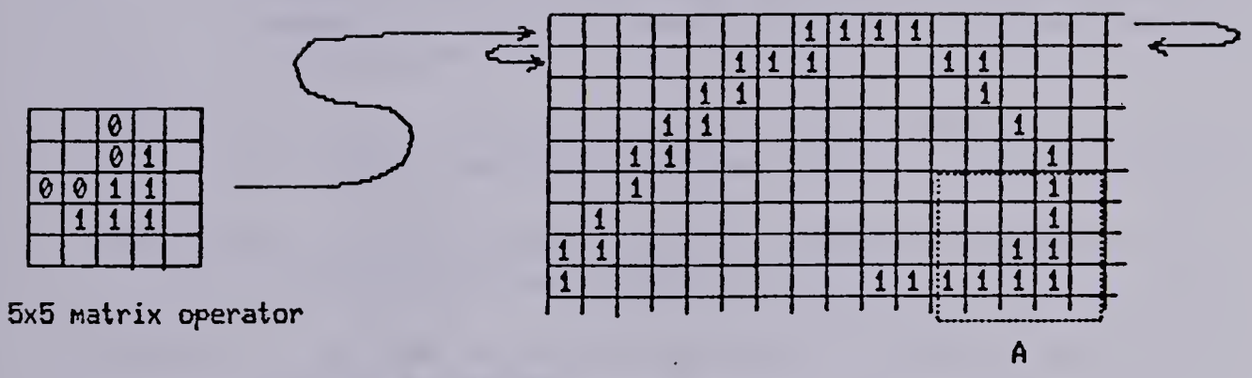


Figure 9 Operator scanning Binary Matrix, with Hit at A

constructed by some combination of two previously calculated operators. In tests with a revised version of their program, Uhr and Vossler reported 100% correct identification on known patterns (a known pattern is one whose list of characteristics has been encountered at least once before) and 96% correct on unknown patterns during the sixth pass through a 26-letter alphabet. The program performed demonstrably better on meaningless patterns than did human subjects (~ 14% error vs. ~ 36% error during fifth pass through the test set).

Currently, one of the predominant approaches to pattern classification is that of Bayesian decision theory. Using the notation of Duda and Hart, the basic technique is as

follows:

Let $\mathbf{X}$ be the vector of features of the image; a features vector.

Let $\Omega = \{w_1,\ldots,w_s\}$ be the finite set of s states which are the possible classifications of the input.

Let $A = \{\alpha_1,\ldots,\alpha_k\}$ be the finite set of k possible actions which are the possible decisions to make, i.e., no classification, classification, rejection, etc.

Let $\lambda(\alpha_i|w_j)$ be the resultant loss for taking action α_i when the state is w_j.

Let $p(\mathbf{X}|w_j)$ be the probability density function for $\mathbf{X}$ given that the state is w_j.

Let $P(w_j)$ be the a priori probability that w_j is the current state.

Then $P(w_j|\mathbf{X})$, which is the a posteriori probability,

$$= \frac{p(\mathbf{X}|w_j)P(w_j)}{p(\mathbf{X})}, \quad p(\mathbf{X}) = \sum_{j=1}^{s} p(\mathbf{X}|w_j)P(w_j)$$

The conditional risk, based on action α_i, is given by

$$R(\alpha_i|\mathbf{X}) = \sum_{j=1}^{s} \lambda(\alpha_i|w_j)P(w_j|\mathbf{X})$$

The Bayes decision rule then is to choose the action $\alpha_i,\{i = 1,\ldots,a\}$ such that $R(\alpha_i|\mathbf{X})$ is minimum.

Although Duda and Hart cite the major difficulty to be that of not knowing the conditional distributions, $p(\mathbf{X}|w_j)$, Kovalevsky (1980, pp. 28-29) claims that the problem lies in its approximate calculation which for "real problems" may require up to $10^{9.5}$ operations to evaluate. Nevertheless, the Bayesian classifier may be used effectively when the input is sufficiently simple in nature.

When both input and target have the same number of components (size of the features vector, same number of edges, etc.), a number of similarity measures which yield values interpretable as "distances" between two patterns may be used. In the following four measures found in Pavlidis (1968), A and B are the image sets or pictures under consideration:

1. $H([A],[B]) = \min_{g \epsilon G} h(A,gB) = \min_{g \epsilon G} h(gA,B)$, where h is the Hausdorff distance, G is a group of isomorphisms. The Hausdorff distance is defined as $h(A,B) = \min \{k | N_k(A) > B \text{ and } N_k(B) > A\}$ where $N_k(A)$ denotes set of all points whose Euclidean distance d from A is less than k, $N_k(A) = \{y | d(y,a) < k \text{ for some } a \epsilon A\}$.

2. $D(A,B) = \sqrt{\sum_{i=1}^{n} (l_i^A - l_i^B)^2}$ is a measure of similarity invariant under translation and is defined for convex polygons with sides parallel to prespecified directions (polygon represented as a sequence of numbers denoting lengths of correspnding sides); l_i is the length of edge i in the polygon, n is the total number of edges.

3. $D'(A,B) = \min_{m} \sqrt{\sum_{i=1}^{n} (l_i^A - m l_i^B)^2}$ is a measure invariant under scale; m is the scaling factor, l_i is the length of edge i in the polygon, n is the total number of edges.

4. $b(A,B) = \dfrac{\text{area } (A \Delta B)}{\text{area } (A \cap B)}$, where $A \Delta B$ denotes symmetric difference of two sets, that is, the set of all points x such that either $x \epsilon A$ and $x \notin B$ or $x \epsilon B$ and $x \notin A$. $b(A,B) = 0$ iff $A = B$. Note that b(A,B) is not a metric whereas the previous measures are.

If, in addition to same component size, the density of the image is high compared to the surrounding area, then clustering analysis may be used. Fu (1982, p. 12) gives a number of commonly used similarity measures (X_i and X_j are the features vectors of two images) that are applicable.

1. Dot product $X_i \cdot X_j = |X_i||X_j|\cos(X_i, X_j)$

2. Similarity ratio $$s(X_i, X_j) = \frac{X_i \cdot X_j}{X_i \cdot X_i + X_j \cdot X_j - X_i \cdot X_j}$$

3. Weighted Euclidean distance
$$d(X_i, X_j) = \sum_{k=1}^{n} w_k (X_{ik} - X_{jk})^2,$$
where w_k are the weights

4. Boolean "and" $\sum_{k=1}^{n} X_{ik} \cap X_{jk}$

5. Normalized correlation $$\frac{X_i \cdot X_j}{\sqrt{(X_i \cdot X_i)(X_j \cdot X_j)}},$$
where $X_i \cdot X_j = \sum_{k=1}^{n} X_{ik} X_{jk}$

Given that the image is represented as some composition of subpatterns or primitives whose relationship may be expressed as a sentence specified by some grammar (tree grammar, string grammar, etc.), then a syntactic approach may be used. For example, given a string representation of target and input images, it is possible to define the distance between these two descriptions by means of the Levenshtein metric (Levenshtein, 1966). Let x and y represent the input and target strings respectively, and σ, γ, and δ be some non-negative weighting factors. Then the weighted Levenshtein distance between x and y is

$$d'(x,y) = \min_j \{\sigma \cdot k_j + \gamma \cdot m_j + \delta \cdot n_j\}$$

where k_j, m_j, and n_j are the number of substitution, deletion and insertion error transformations.

Another example of syntactic pattern recognition, using a tree grammar system, as applied to classification of bubble chamber events and some arbitrary English characters is found in Fu and Bhargava (1973). Their techniques produced error rates of four percent or less on preprocessed bubble chamber data.

By design and necessity, many significant algorithms, techniques, and experiments applicable to pattern classification have been omitted. The diversity of approaches taken and their variations are not sufficiently covered by any known single reference. However, for the interested, Fu (1982) contains a wealth of information while Sampson (1976) provides an especially nice but concise historical perspective.

It is anticipated that the reader will have by now gained some sense of the frustration and difficulties faced by researchers and practitioners in this area. Not only are the applications wide-ranged but the diversity and permutations of techniques have caused some to suggest a standardization, a formalization of pattern recognition into a language similar to that of the predicate calculas. Simon (1975) has called the current practice a "bag of tricks" approach, but the bag of tricks approach is fundamental to

the operation of GMATCH. As will be seen in the next chapter, GMATCH borrows heavily from template-matching, matrix operators, sequential scanning, and moments analysis. In fact GMATCH is designed in a fashion that permits the bag to grow, that is, it has a structure that permits incorporation of more layers of algorithms to solve different matching problems.

III. Formal Specifications for Graphics Input and Analysis

A. System Overview

A CAI system employing the GMATCH procedure is represented in Figure 10. The dashed boundary indicates the author interface while the dotted boundary indicates the student interface. It is evident from the diagram that GMATCH proper (horizontally-shaded region) is a subset of the CAI system and that a major subset of GMATCH (cross-hatched region) is also a subset of the student environment (vertically-shaded region).

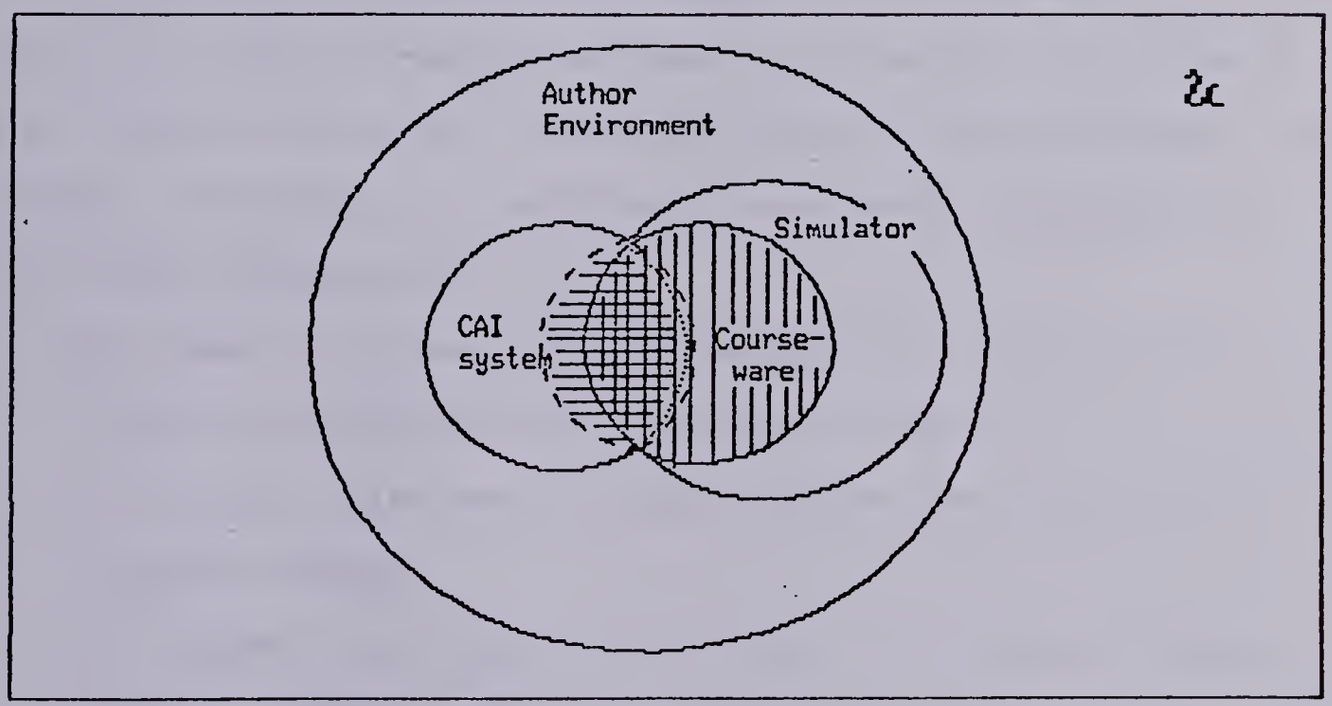


Figure 10 Venn representatton of system components

The author environment is seen to be a superset of the CAI system.

The authoring environment contains four major components:

1. a language or process that produces both source and object code for courseware production,
2. a simulator to allow the author to fully test the generated courseware,
3. other system tools and supporting utilities including file editors, character set editors, sort routines, performance record programs, etc., and
4. the GMATCH procedure and supporting environment.

The first three of these components are beyond the scope of this study, hence, their features will not be described. The GMATCH environment, as previously mentioned, consists of four major components:

1. the data structures and the subsequent syntax of the language extensions for controlling GMATCH,
2. the author interface for the creation and testing of graphics images,
3. the student interface for the input of graphics images within a course, and
4. the GMATCH analysis routines proper.

B. Data Structures and Language Extensions

Data structures

The notation employed in the following definitions (Davis, 1982) is summarized as follows:

a. The primary building blocks or modules of the description language used here are called **productions.**

b. A required production is enclosed in < >, while an optional production is enclosed in [ ].

c. A required production that is repeated n times is shown as <n: production_name>

d. An optional production that may be repeated from Ø to n times is [n: production_name]

e. An optional repeating production is shown as [production_name]...

f. Strings are enclosed in quotation marks as in "0b", while comments are shown as /* comment string */.

g. The construction *type *range (n,m) indicates data of type integer or floating point within the range n to m inclusive.

```
Definition:  <coord_pair>
                    is          *integer   *range(0,32767)
                                          /* x-coordinate */
                                ","
                                *integer   *range(0,32767)
                                          /* y-coordinate */
                    ;

Definition:  <more_coord_pair>
                    is          ";"
                                <coord_pair>
                    ;
```

i) Image

An image is any set of objects which may be displayed at any one time.

```
<image>
        is    1 to 50 of <object>
        ;
```

ii) Object

An object is a set of primitives of the following types: point, vector, arc, circle, triangle, rectangle, and polygon (future extension of this list is possible). Objects of an image are consecutively numbered up to a maximum of 50 at any one time. For the purposes of recognition or matching, the null object is not defined.

```
<object>
        is          "Ob"
                    *integer *range(1,50)
                         /* object number */
                    " "
                1 to 50 of <primitive>
        ;

<primitive>
        is          <point>
        or          <vector>
        or          <arc>
        or          <circle>
        or          <triangle>
        or          <rectangle>
        or          <polygon>
        ;
```

iii) Point

A point is the simplest primitive, visualized as a single dot (pixel) and specified by a single pair of Cartesian coordinates.

```
<point>
        is          "d"
                    [RGB_triple] /* color */
                    <coord_pair>
        ;
```

```
<RGB_triple>
        is          "["
                        /* see Appendix B */
                    *float *range(0,1)
                        /* Red */
                    ","
                    *float *range(0,1)
                        /* Green */
                    ","
                    *float *range(0,1)
                        /* Blue */
                    "]"
                ;
```

iv) Vector

A vector is visualized as a line segment extending from a beginning point to a terminating point inclusive.

```
<vector>
        is          "v"
                    [RGB_triple]
                    <coord_pair>
                        /* begin point */
                    <more_coord_pair>
                        /* end point */
                ;
```

v) Arc

An arc is visualized as a curvilinear segment of a circle and is specified by three pairs of Cartesian coordinates representing the starting point, a point along the curvilinear segment, and the ending point respectively.

```
<arc>
        is          "a"
                    [RGB_triple]
                    <coord_pair>
                        /* begin point */
                    <2:more_coord_pair>
                        /* point, end
                        point */
                ;
```

vi) Circle

A circle is specified by the coordinates of the centre and its radius. A circle with a radius of 0 is functionally equivalent to a point.

```
<circle>
        is          "c"
                    [RGB_triple]
                    <coord_pair> /* center */
                    ","
                    *integer *range(0,16000)
                         /* radius */
                ;
```

vii) Triangle

A triangle is visualized as a closed figure composed of three vectors. It is specified by three Cartesian coordinate pairs representing the locations of its vertices.

```
<triangle>
        is          "t"
                    [RGB_triple]
                    <coord_pair>
                         /* vertex 1 */
                    <2:more_coord_pair>
                         /* vertices 2,3 */
                ;
```

viii) Rectangle

A rectangle is a closed figure composed of four vectors such that the end points of pairs of adjacent vectors meet at right angles. The specification of a rectangle requires any two diagonally opposite vertices.

```
<rectangle>>
        is          "r"
                    [RGB_triple]
                    <coord_pair>
                         /* 1st vertex */
                    <more _coord_pair>
                         /* diag. opp.
                         vertex */
                ;
```

ix) Polygon

A polygon is a closed figure composed of n vectors, where n => 4. The specification of a polygon requires n coordinate pairs for the n vertices.

```
<polygon>
        is          "p"
                    [RGB_triple]
                    <coord_pair>
                         /* begin vertex */
                    <3:more_coord_pair>
                         /* others */
                    [more_coord_pair]...
        ;
```

Language extension syntax

To control the loading and display of target images as well as student input images, several new language commands have been defined. These commands are intended as extensions to the graphics sub-language of any CAI language or system.

```
Definition:  <filename>
        is          [directory_name]
                    <alphabetic>
                    [19: alphanumeric]
                    [extension]
        ;

Definition:  <directory_name>
        is          "["
                    <alphabetic>
                    [19: alphanumeric]
                    ".dir]"
        ;

Definition:  <alphabetic>
        is      1 of "abcdefghijklmnopqrstuvwxy
                     zABCDEFGHIJKLMNOPQRSTUVWXY
                     Z"
        ;
```

```
Definition:  <alphanumeric>
                    is          "abcdefghijklmnopqrstuvwxy
                                zABCDEFGHIJKLMNOPQRSTUVWXY
                                Z0123456789_"
                    ;

Definition:  <extension>
                    is
                         1 to 3 of <alphabetic>
                         excluding "dir"
                    ;
```

i) Load graphics target

The load graphics target command assigns as the target variable array those objects from each target image file specified in the target list (<target_list>). The target list consists of the names of the image files on disk and their associated object lists (similar to records of random access text files). The target list can also consist of a list of string descriptors (<prim_list>) in which case the targets are the general classes of objects such as triangles, rectangles, hexagons, etc. rather than specific objects. Defaulting the object list causes all objects in the given image file to be assigned to the target variable array. In conjunction, the corresponding data structures for the visual display of the targets is automatically assigned to a display list structure.

```
<load_graphics_target>
          is          <load_code>
             1 to 5 of " "
                      <target_list>
          ;

<load_code>
          is          "lgt"
          or          "LGT"
          ;

<target_list>
          is          <file_list>
          or          <string_list>
          ;

<file_list>
          is          <file_object>
                      [4:more_file_object]
          ;
```

```
<more_file_object>
        is          ","
                    <file_object>
        ;

<file_object>
        is          <filename>
                    [objects]
        ;

<objects>
        is          ":"
                    <obj_num>
                    /* object numbers */
                    [49:more_obj_num]
        ;

<obj_num>
        is          *integer *range(1,50)
        ;

<more_obj_num>
        is          ","
                    <obj_num>
        ;

<string_list>
        is          <prim_list>
                    [49:more_prim_list]
        ;

<more_prim_list>
        is          ","
                    <prim_list>
        ;

<prim_list>
        is "point" or "vector" or "arc" or
           "circle" or "triangle" or
           "rectangle" or "square" or
           "rhombus " or "trapezoid" or
           "quadrilateral" or "pentagon" or
           "hexagon" or "heptagon" or
           "octagon" or "nonagon" or
           "decagon"
        ;
```

Note that the <prim_list> can be altered at any time by the author to include new object names or delete existing ones. The specification will require pertinent identifiable features to distinguish the different classes of objects. For example, a triangle is identified as having three vectors such that

they form a closed object, that is, the coordinates of the end point of one vector consequently becomes the coordinates of the start point of the succeeding vector. However, once these features have been defined, they become transparent to the author and are automatically linked during compilation of the source code if their names are found in the <string_list>.

ii) Window

The term window is used here to describe any rectangular region on the display screen or CRT. Technically speaking, the proper term is viewport (Foley & Van Dam, 1982, pp.40-42). The window command creates a window on the existing screen display, causing the specified area to be cleared of any image (erased). The window can be set with an existing background color and/or with a colored border. It controls the area for the display graphics target command and also the graphics input command.

```
<window>
          is              <window_code>
            1 to 5 of " "
                          <border>
                          <background>
                          <coord_pair> /* upper left
                               corner */
                          <more_coord_pair> /* lower
                               right corner */
          ;

<window_code>
          is              "win"
          or              "WIN"
          ;

<border>
          is              "o" /* outline border */
                          <RGB_triple>
          or              "n"  /* no border */
          ;

<background>
          is              "b" /* background */
                          <RGB_triple>
          or              "n"  /* no background
                               (black) */
          ;
```

iii) Display graphics target

The display graphics target command causes the specified objects to be displayed on the monitor screen according to the specifications of the transformation list. For example, suppose object 1 is a circle located at coordinates 600,300 and it is desired to place it at 200,150 then the transformation list must indicate a translation of -400,-150 (relative to current location). Defaulting the object specification results in all objects in the <target_list> being displayed. Defaulting the transformation list implies no transformation. The specified objects must, of course, be one or more of the objects specified by the load graphics target command otherwise a a compile-time error indicating inconsistent specification of objects will be generated. Objects which do not fit entirely within a designated window are automatically clipped during display, that is, that portion of an object existing outside the window boundaries, is not displayed. Note that this is a display feature only and does not affect the data structure of the object.

```
<display_graphics_target>
        is          <display_code>
          1 to 5 of " "
                    <window_spec>
                    [object_list]
        ;

<display_code>
        is          "dgt"
        or          "DGT"
        ;

<window_spec>
        is          "l" /* left half of window
                         */
        or          "r" /* right half of
                         window */
        or          "a" /* all of window */
        ;
```

```
<object_list>
        is          [transform_list]
                        /* transformations */
                    *integer *range(1,50)
                        /* names of objects
                        operated on by
                        transform _list */
                    [49:more_object]
        ;

<more_object>
        is          ","
                    [transform_list]
                    *integer *range(1,50)
        ;

<transform_list>
        is          "["
                    <translation>
                    ","
                    <rotation>
                    ","
                    <scale>
                    "]"
        ;

<translation>
        is          <displacement>
                        /* signed pair */
        ;

<rotation>
        is          *integer *range(0,359)
                        /* degrees of
                        rotation */
        ;

<scale>
        is          *float *range(.1,20)
                        /* scaling */
        ;

<displacement>
        is          *integer *range
                        (-32766,32767)
                    ","
                    *integer *range
                        (-32766,32767)
        ;
```

iv) Graphics input

This command results in the activation of the user interface routines to permit the

subsequent input of a graphics image by means of a graphics tablet, lightpen or mouse. The window specifier allows the input to be limited to the left-half of the last defined window, to the right-half of the window or to all of the window area. The time parameter allows the author to control the amount of time permitted for the entry of any graphics image. Defaulting this parameter permits the student to have unlimited time to respond. The drawing function parameter controls the different drawing primitives available for use by the student. Defaulting the drawing function parameter permits the student to have full access to all available drawing primitives. Currently, only seven drawing primitives have been defined. These are vector, continuous vector, line, rectangle, arc, circle, and draw. They are referenced by integer values in the range of 1 to 7 respectively (see the Examples Section following). Allowance has been provided for extension of these primitives to include up to 255 different types.

```
<graphics_input>
        is              "gin"
          1 to 5 of     " "
                        <window_spec>
                        [parm]
        ;

<parm>
        is              <time> /* input time
                             control */
                        ";"
                        <draw_prim> /* drawing
                             primitive */
        or              ";"
                        <draw_prim>
        or              <time>
        ;

<time>
        is              *integer *range(1,3600)
                             /* seconds */
        ;

<draw_prim>
        is
                        *integer *range(1,255)
                          /* only 7 currently
                             defined */
                        [6:more_integers]
        ;
```

```
<more_integers>
        is          ","
                    *integer *range(1,255)
        ;
```

v) Graphics analysis

This command invokes the analysis of student input. Input of graphics images does not automatically invoke the analysis of said images upon the termination of the input. Analysis of graphics images occurs only when the graphics analysis (gan) command is explicitly stated. This format allows the author to permit students to have "doodling" access without subsequent analysis occurring. The gan command has an optional parameter list which controls the following aspects:

1. limitation of time for analysis,
2. comparison of student input against the transformed target, and
3. analysis with report (student feedback on performance) or with no report.

Defaulting the parameter list provides for unlimited time for analysis (actually, limited only by system time limit), no report, no transformation list, and complete hierarchical access.

```
<graphics_analysis>
        is          "gan"
                    [gan_parm_list]
        ;

<gan_parm_list>
        is
          1 to 5 of " "
                    <parm2>
        ;

<parm2>
        is          <time>
                    [50:Otr]
                    [R]
        or          <50:object_transform_list>
                    [R]
        or          <report>
        ;
```

```
<Otr>
        is          ";"
                    <object_transform_list>
        ;

<object_transform_list>
        is          <transform_list>
                    /* transformations */
                    *integer *range(1,50)
                    /* object name */

<R>
        is          ";"
                    <report>
        ;


<report>
        is          "r" /* report */
        or          "n" /* no report */
        ;
```

Examples

The following three examples illustrate the versatility of the commands governing the use of GMATCH. Example 1 shows the simplest possible structure, Example 2 shows a more complex arrangement, while Example 3 is considerably more complex.

Example 1

Suppose an instructor using a CAI system with GMATCH wishes to test a child's comprehension of the concept of triangle. A sufficient test might require the child to draw a triangle using a graphics tablet for input. However, the instructor obviously would not want to display any image to the student while posing the question, hence, the dgt command is **not** required.

In this example, the load graphics target command loads a single target image from a file called "triangle.reg" which contains the single string primitive "triangle". Graphics input is only limited to the predefined window area and complete hierarchical analysis can occur without restrictions.

```
lgt  triangle.reg

win  nn 25,351;600,550
          /* input window set at upper-left
             coordinate of 25,351 and lower-
             right coordinate at 650,550 */

gin  a    /*  use all of window for input */

gan       /*  analyze without restriction */
```

Example 2

A teacher wishes to design a question to test student performance on motion geometry, in particular translation and rotation of a given object. Rather than creating a new image, a satisfactory object is found in an existing image file called "quad.reg", which contains three objects - a parallelogram, a trapezoid and a square, labelled 1 to 3 respectively. The desired object, a trapezoid (object 2), is currently situated at location (600,100) but the desired location is at (50,250), which is in the left half of a window area. Hence, the following commands are required:

```
lgt  quad.reg:2  /* load the trapezoid */

win  o[0,1,0]n0,100;767,399
                 /* create window with green border,
                    no background and with dimensions
                    768 x 300 starting at 0,100 */
```

```
dgt  l[-550,+150,0,1]1
                    /* perform translation, no rotation,
                       no scaling on the only object
                       loaded - now called object 1 */
```

The task for the student is to recreate the object on the right half of the window but with a translation of 50 pixels to the right and rotated 1/8 turn counterclockwise. A time limit of 2 minutes using only vector drawing mode is permitted. No time limit is specified for the analysis, but a report is desired. Here the gin and gan commands will be:

```
gin  r120;1        /* right half of window for input;
                      2 minute response limit and drawing
                      mode 1 (vector) */

gan [+50,0,45,1]1;r
                    /* 50 pixel translation to the right;
                       45° counterclockwise rotation;
                       no scaling on object 1 and report
                       */
```

Example 3

This example demonstrates a complex load arrangement which loads different objects from different image files (Figure 11). Some of these are then displayed within the left-half of a bordered window with no background color. Graphics input is limited to 180 seconds (3 minutes), right-half of the window, and 4 drawing primitives (vector, continuous vector, line, and draw). Analysis is performed on the translated targets with report and a 2 minute time limit.

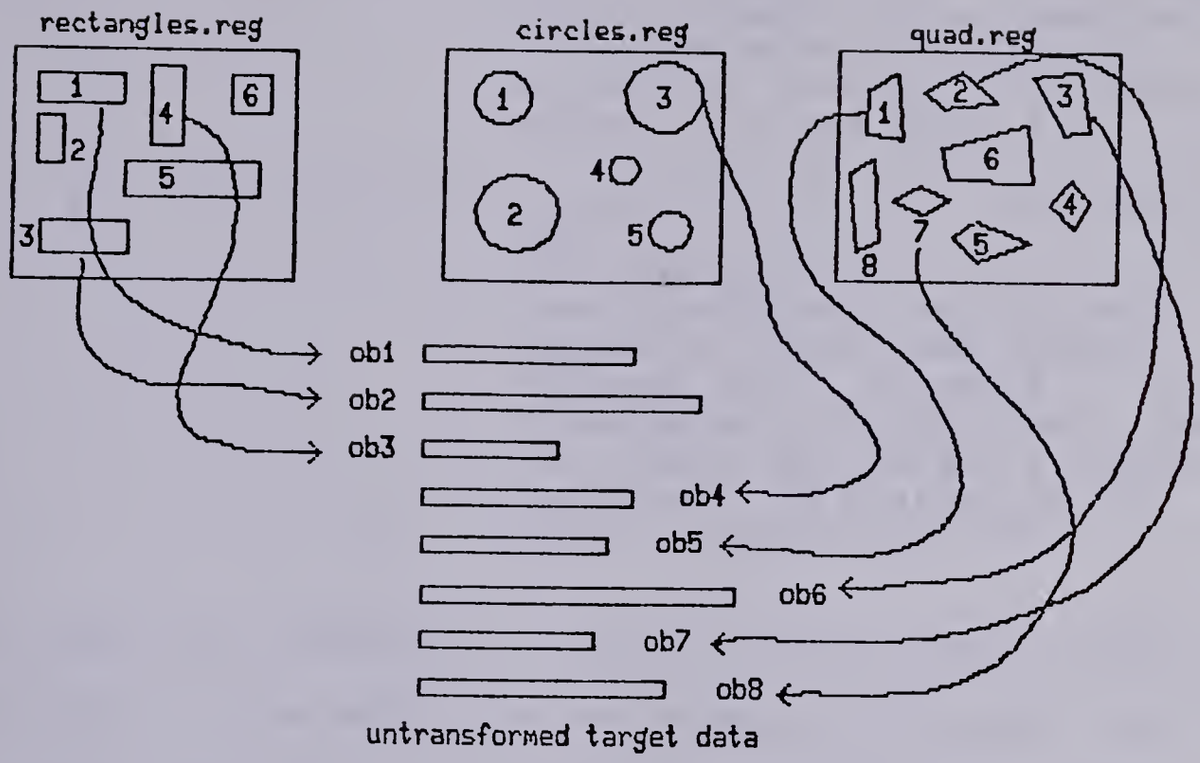


Figure 11 Loading Objects from Three Files

```
lgt  rectangles.reg:1,3,4;circles.reg:3;quad.reg:1,2,3,
                    7
                 /* once loaded, objects are referred
                    by numbers 1-8 respectively */

win  o[0,1,0]n0,100;767,479
                 /* green border, black background,
                    full width (768 pixels), partial
                    height (380 pixels) window */

dgt  l[0,+50,0,1]1;[-300,+200,0,.8]4;8
                 /* display objects in left-half of
                    window (clipping in effect). Object
                    1 is translated down from its
                    current position by 50 pixels, no
                    rotation, normal size.  Object 4 is
                    translated left by 300 pixels and
                    down by 200 pixel, no rotation and
                    scaled to .8 of its normal size.
                    Object 8 is displayed unchanged
                    from its original specs. */
```

```
gin  r180;1,2,4,7
                    /* input on right-half of previously
                       defined window with a time limit of
                       3 minutes using only drawing
                       primitives:  vector, continuous
                       vector, line and draw. */

gan  120;[0,0,135,1]1;[+200,+150,0,1]4;[-100,-100,180,
                       1]8;r
                    /* 2 minute analysis limit,
                       input graphics to be compared
                       against a transformed target 1
                       (rotated 135°), target 4
                       (translated left 200 pixels, down
                       150 pixels) and target 8 (trans-
                       lated left 100 pixels, up 100
                       pixels, and rotated 180° */
```

The power and complexity of these commands would normally be difficult to use except for the simplest of cases, since the parameter list as seen in the lgt, dgt and gan commands in Example 3 can be quite lengthy. An author would have a difficult time recalling the proper syntax and the associated transformation and object lists for these commmands. Hence, there is a need for automatic generation of parameters (code generator) through the interactive author interface for GMATCH, which is described in the following section.

C. Author Interface

The author interface consists of the requisite hardware and software to enable one to create new graphics targets (objects, images), load existing targets, test transformation parameters and analysis routines, and generate GMATCH commands. A graphics tablet, lightpen, mouse or touch-sensitive screen can be the designated means of

entering coordinate data for the creation of graphics images, although the tablet is probably most versatile for this application. The authoring terminal and host system should be able to support visual editing, multi-tasking with defineable windows, and have at least three bitplanes. A multi-tasking operating system would permit compilation of a program module in one window, execution of another module in a different window and graphics editing within a third window, while multiple bitplanes would provide an extremely fast and relatively simple means by which to manipulate different graphics objects or images independently of one another. For example, bitplane one can be used for the target image, bitplane two for the input image and bitplane three can be used for the various menu displays for both author and user interfaces. Additional bitplanes would enable independent color manipulation of objects. A prototypal author interface display illustrating windowing capabilities and menus (Figure 12.) is described below.

Status:
This line of the display indicates the current drawing status. In this example, object 2 is being created in green using the vector drawing function.

Drawing function:
At the bottom of the display, a list of various drawing functions and patterns is shown. The author can use the pen cursor to select the function and pattern of choice. The

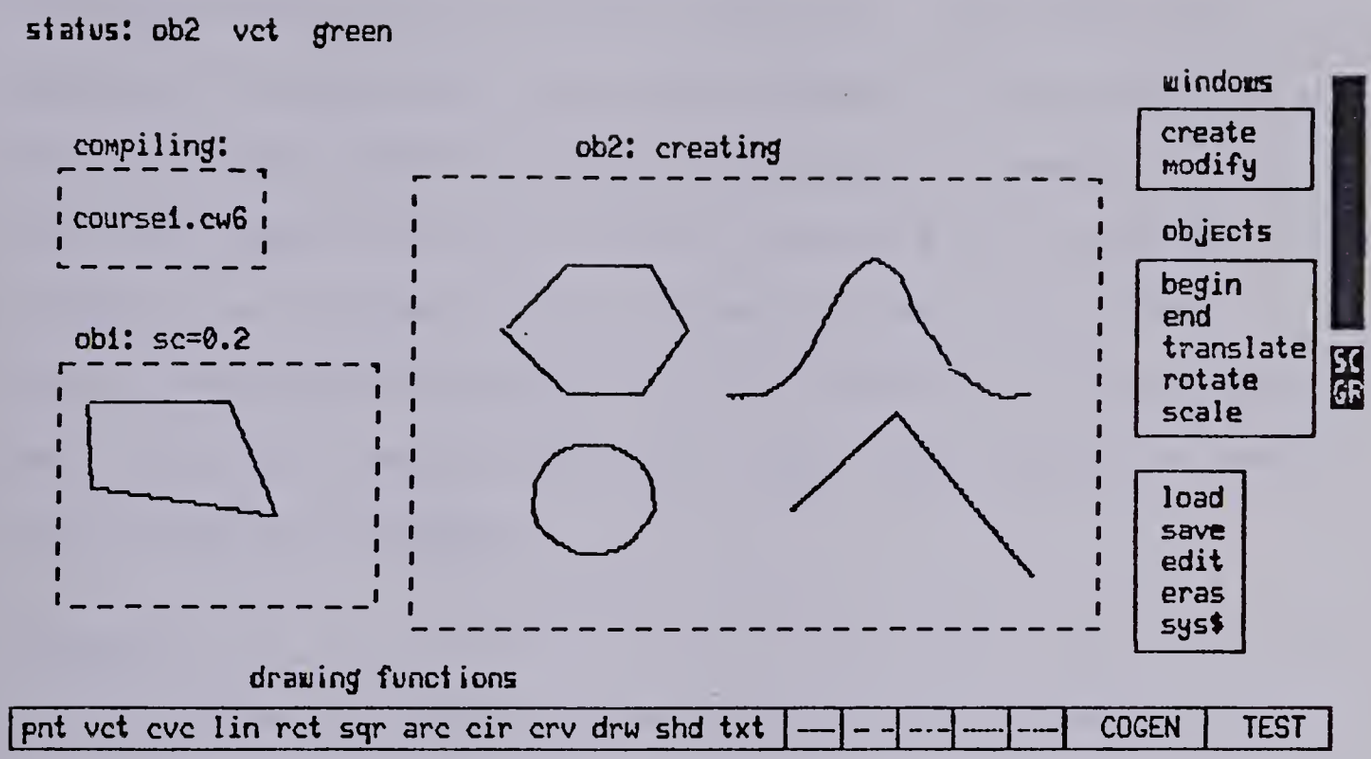


Figure 12 Prototype Author Interface Display

abbreviations represent the following: point, vector, continuous vector, line, rectangle, square, arc, circle, curve, draw, shading and text, respectively. The vector, continuous vector and line drawing functions generate data structures of type vector with these distinctions. The continuous vector function produces a sequence of vectors such that the end point of one vector is subsequently the start point of the next vector. However, if the object being created is closed then the data structure will be either triangle or polygon. Both square and rectangle create data structures of type rectangle. Both curve and draw create sequences of type point which are digitally-connected (Kim, 1982). The shading function does not generate any data type

or modifier per se but provides a means for the author to highlight particular objects. Similarly, text does not generate a unique data type (see Chapter 5) but provides a means for the author to insert descriptive comments into both the image files for pattern matching and those for display, or to create text windows as part of a screen image. The drawing patterns do not affect any of the data structures for pattern matching; they only affect the data structures for display.

Windows:

At the right hand side of the display are four boxes. The top box, which is labelled "windows", permits the author to create viewports (open windows). Each viewport can contain or display different activities or objects or images. For example, the screen display contains one window in which a compilation of a CAI module is occurring. Another window shows an object labelled as "ob1" scaled down to 0.2 of its real size, while the current window is labelled "ob2: creating". The windows can be modified by scaling, translating, deleting, coloring or temporarily hiding. A maximum of twenty windows can be opened at any one time.

Objects:

An object definition is invoked through the begin command in the objects box and is terminated by the end command. As mentioned under Data Structures (Section B), an object is any set of primitives. Objects, whether being created or

loaded from previously created image files, can be translated, rotated, or scaled within the current window. Note that automatic clipping is performed on all objects which exceed the bounds of the window coordinates.

Unlabelled:

The third box down on the right hand side contains five commands: load, save, edit, eras, and sys$. The load command permits the loading of complete image files or objects within image files or program modules into the specified window. The save commmand causes objects or program modules within the current window or any other specified window to be saved. Edit permits the alteration of an object within the current window or the alteration of a file. The eras command allows one to erase (delete) the last primitive, the current object or any specified object or file. The sys$ command allows the author to directly enter a system level command (such as purging files, checking directories, etc.) without leaving the current environment. Near the bottom right corner, the words "COGEN" and "TEST" appear. Invoking the COGEN command causes a menu (Figure 13) for the code generator to appear. Recall that the code generator permits easy construction of parameters for GMATCH commands. This is accomplished by maintaining the lists of commands associated with every window creation. Once a GMATCH module has been created, the author can immediately execute and verify the code through the TEST command without having to leave the current environment. At the far right hand side of the

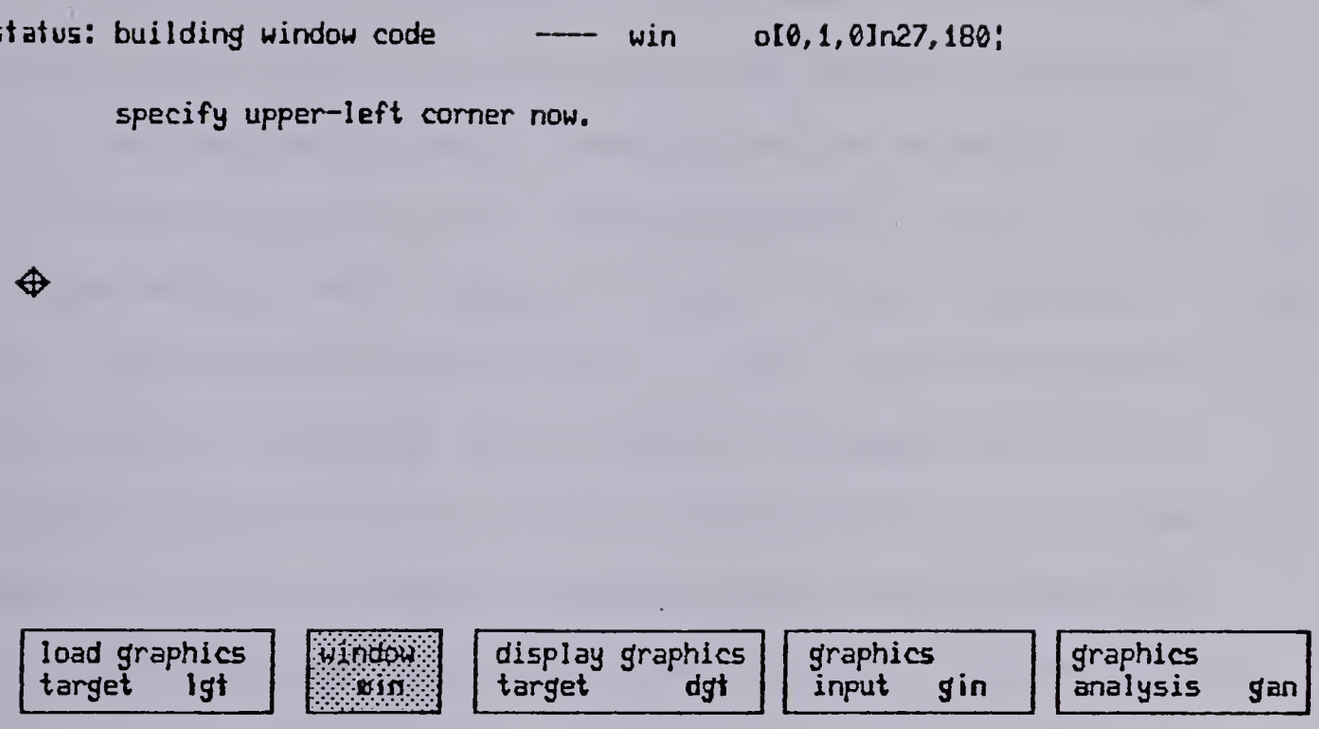


Figure 13 COGEN Menu Display

display, a black vertical bar is shown. On a color monitor, this bar is seen as a collection of eight, small, colored rectangles representing drawing color selections. Terminals capable of supporting more colors would have an auxiliary color chart from which additional colors could be chosen. Directly below this are two small rectangles containing inverse-lettering. The upper one (SC) controls background screen coloring for windows, while the lower one (GR) places or removes a grid onto the current window. This permits the author to finely position objects within a window. Selection of color, background screen color, or grid is made by moving the pen cursor over the desired choice and momentarily depressing it.

D. User interface

The hardware component of the user interface is the same as that for the author with one notable exception, a terminal supporting multi-tasking is not essential since it is assumed that the only task per student is the current CAI course module. With respect to the software component of the user interface, it lacks many of the functions present in the author interface, furthermore, its appearance will change since it is, in effect, controlled by the author's specification of GMATCH commands. Both the win and gin commands directly affect the user display. Figure 14 illustrates one possible configuration for the question shown. Note that the drawing functions have been limited to vector, continuous vector, line, triangle and polygon. Also note that the user can place a reference grid over the entire window area by touching the pen cursor to the location marked "GR". The rectangle marked "end object" allows the user to terminate an object definition and to begin a new definition or to terminate an object definition and leave input mode. This automatically invokes the analysis phase, which is the subject of the next section.

E. GMATCH HIERARCHY

GMATCH overview

In his introduction to Digital Picture Analysis, Rosenfeld (1976) identified the four goals of pictorial

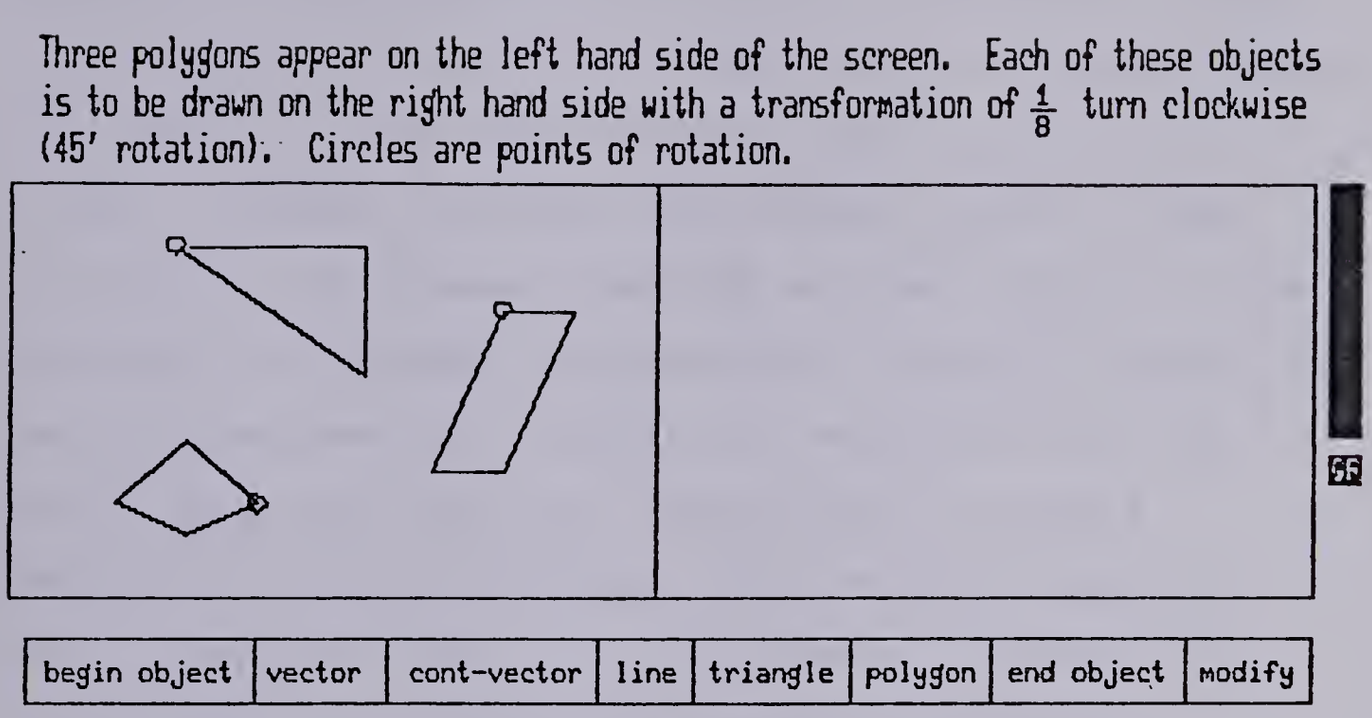


Figure 14 Example of User Interface Display

pattern (image) recognition as being:

1. matching pictures with one another,
2. classifying a picture into one or more sets of classes based upon properties or features of the picture,
3. segmenting a picture into components and classifying or describing these components, and
4. recognizing a picture's components in relation to each other.

Although perhaps not to the extent implied by Rosenfeld, these are also the goals of the GMATCH hierarchy, which consists of an entrant decision matrix and a five level hierarchy of analysis routines.

The current specification of the GMATCH analysis routines consists of a five level structure with expansion

capabilities up to sixteen levels. The levels are graded according to the degree of specificity of operation, depth of analysis, and complexity of code. Level I analysis checks for matching of the primitive data types: point, vector, circle, triangle, rectangle, and polygon. Level II adds collinearity and closure checking on input vectors for the existence of polygons, decomposition of target polygons into vector primitives, and analysis for equivalent arcs and string description analysis, while Level III uses a sequential scanning technique with a weighted operator. Level IV applies moment analysis to sets of points and, unlike Level III analysis, is capable of similarity matching. Lastly, Level V provides a similarity analysis on polygonal objects.

GMATCH control

In its unconstrained, but ordered operation, sets of objects are allowed to systematically sift thorugh all the levels of GMATCH until the set of input objects has been exhausted or until all levels of the hierarchy have attempted to match or recognize the objects. It should be obvious even to the initiate in this area that such a procedure, while possibly possessing a high degree of success in matching if matching is possible, is burdensome. What is desirable is a decision-making process which bypasses those levels of the hierarchy which have no (or very low) possibility of success in matching the objects. A

solution is achieved by setting up two control vectors. The first vector (target control vector) is the result of the constraint imposed by the target data type specifications. The second vector (input control vector) arises from a fast parsing of the input to determine the nature of the input data types. Together, these two vectors define the entry into a decision-matrix (Figure 15) which ultimately controls entry into the GMATCH analysis hierarchy.

Typically, the procedure is as follows.

1. During compilation, the target objects are parsed to determine their data types. A target object data type list (target control vector, **tc**) is then created with an internal ordering of data types for each object.
2. As each student input object is defined, its data types are maintained in an ordered input object data type list called the input control vector, **ic**.
3. Both target data and input data are maintained as arrays. Copies of both are initially made before any analysis begins. Analysis is subsequently performed on these copies, henceforth referred to as **T** and **I** respectively.
4. The **ic** is parsed in an attempt to order the input objects (using the data type specifications) in the same sequence as that appearing in the **tc**. At this stage it is not possible to ascertain whether or not the input object data and the target data are referring to the same objects, only that the data types are compatible

STUDENT INPUT

	vct cvc lin drw	vct cvc lin	arc	circle	cvc	cvc rectangle	cvc
	•	⟋	⌒	○	△	▭	⛰
•	I III IV	III	III IV	III IV	III IV	III IV	III IV
⟋	III IV	I II	III IV	IV	II V	II V	II V
⌒	III IV	III IV	I II	III IV			
○	III IV	III IV	III IV	I			
△	III IV	II V			I II	II V	II V
▭	III IV	II V			II V	I II	II V
⛰	III IV	II V			II V	II V	I II

Figure 15 Decision Matrix for GMATCH Entry

(in this sense, compatibility means that an nonempty entry exists within the decision matrix). To increase the probability that the comparison between input object and target object is the desired comparison (i.e. referencing the same object of an image), two switches and a pointer list are created for each input object. Switch 1 is the data type (primitive) count switch, s1, and switch 2, s2, is the alternative object switch. If the number and type of primitives found in the input object matches that found in the target object, then s1 is turned on (on = 1, off = 0). The same input object is then checked against the remaining target objects for

data types and their total number. If any other target objects are found that match on these two parameters, then s2 is turned on and the pointer list will point to the locations of the alternative objects in **T**. An arbitrary example of this mechanism is shown in Figure 16 in which an input set and a target set consist of three objects each. The target control vector is shown to be

tc = |vector,arc|triangle|rectangle|,

where the "|" symbol serves as a delimiter between objects. Similarly, the original **ic** is

ic = |vector|vector,arc|vector|.

A reordering of the first two objects in the **ic** gives a better match with the **tc**, namely

ic = |vector,arc|vector|vector|.

For the first object then, both **ic** and **tc** match with respect to primitive type, hence, s1 = 1. S2 = 0 since no other object in **tc** contains the same data types. For objects 2 and 3, the data types in **ic** do not match any of those found for objects 2 and 3 of the **tc**, hence, s1 and s2 = 0 for both objects.

5. If step 4 has been successful (every input object has a corresponding target object composed of the same data types), then the decision rules for analysis are specified by the entries in the main diagonal of the decision matrix, i.e., Decisions = $\{d_{i,j}\}$, where i=j. If step 4 is unsuccessful, then the decision rules are

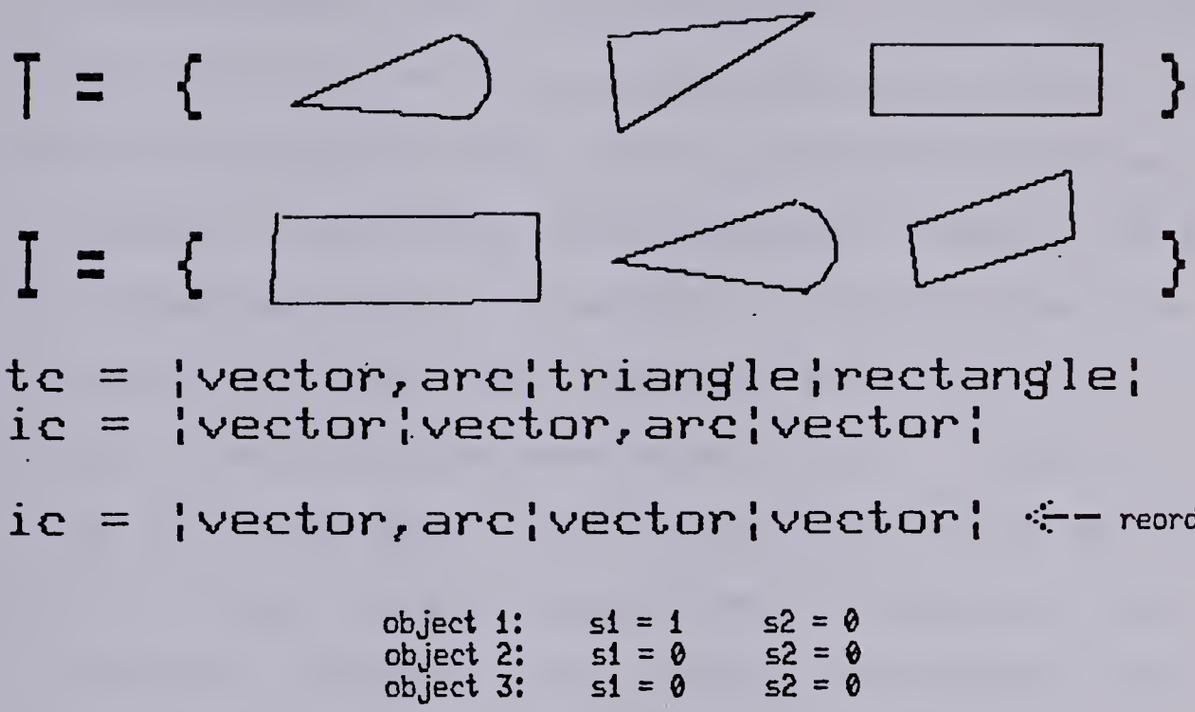


Figure 16 Generating Control Vectors and Setting Switches

specified by the entries above the main diagonal (i < j) or below the main diagonal (i > j). Note that most cells above the main diagonal (shaded cells) represent highly unlikely scenarios, the assumption being that an author will choose the most convenient and elegant way for expressing an image. For example, if an author wishes to have an arc as the target object, it would be highly unlikely and most inefficient for him or her to specify that arc using the point data type. Hence, the student input of "arc" and target data type "point" represent an unlikely situation but which can be analyzed at levels III and IV.

6. Each object, therefore, will have a set of decision

rules governing the levels of the hierarchy to be used, with the lower levels applied first. A successful match will result in the input object data being moved into the **correct-match array** and the corresponding data in T deleted. If the match is unsuccessful, then s1 and s2 are examined. If s1 = 1 and s2 = 0, then there is a strong possibility that the match is wrong since no other likely alternatives exist. If s1 = 1 and s2 = 1, then the input object must be checked against the other likely target objects identified in the pointer list before any judgement can by made as to whether the input is incorrect. If s1 = 0 and s2 = 1 or s1 = 0 and s2 = 0, then there is a strong possibility that the current objects being compared are not the desired comparisons and that the other objects specified by the pointer list must be checked out. An object judged to be wrong is then moved into the **wrong-match array.**

Once the control vectors have been set up and the decision of which levels of the hierarchy to access is made, program flow jumps directly to the lowest of the possible hierarchy levels. As an example, consider the case of input and target objects employed in Figure 16. For object 1 which consists of vector and arc data types the decision matrix indicates that the vector should be analyzed by level I first and if there is no match, then by level II. For object 2 whose input data type is vector and target data type is triangle, the matrix shows level II analysis first and then,

if necessary, level V analysis. Finally, for object III whose input type is vector and target type is rectangle, the matrix again specifies level II analysis followed by level V analysis.

The next several sections describe in detail the algorithms employed for analysis.

NOTATION

The following notation is employed.

T is the duplicate set of transformed targets, that is
T = {transformations} x {targets},
$T = \{t_1, t_2, \ldots, t_n\}$

I is the duplicate set of input objects,
$I = \{i_1, i_2, \ldots, i_m\}$.

i is the current input object.

t is the current target object.

{ip} is the set of primitives (synonomous with data types) for the current input object being analyzed.

ip is the input primitive currently under consideration.

type:ip is the 'type' of the ip (for example, vector).

parm:ip is the set of parameters associated with a particular ip.

{tp} is the set of primitives for the current target object.

tp is the target primitive under consideration

type:tp is the 'type' of the tp.

parm:tp is the set of parameters of the tp.

Pci(n) is the array of primitives that have been correctly matched for the current input object.

Pui(n) is the array of primitives that have failed to match on the first pass (unmatched) for the current input object.

Pwi(n) is the array of primitives that have been identified as being incorrect for the current input object.

Pct(n) is the array of primitives of a target object that have been matched.

Put(n) is the array of primitives of a target that are unmatched.

CMi(n) is the correct-match array (array of correctly-matched objects).

WMi(n) is the wrong-match array (array of incorrectly-matched objects).

ϵ is the tolerance (specified as +/- n pixels or points).

<> is not equal to (complete failure)

~<> is partially not equal to (partial success;partial failure) as in parm:ip ~<> parm:tp means that some, but not all, of the parameters within the parameter lists matches. For example, matching two out of three coordinate pairs which specify the vertices of a triangle.

F. Level I Analysis

Error Limits

Level I analysis is essentially a congruence detection algorithm in which every primitive from an input object must be matched against a corresponding primitive from a target object within a prespecified tolerance. For example, an input point is said to match a target point if and only if

$$(x,y)_{input} = (x \pm \epsilon, y \pm \epsilon)_{target}.$$

The tolerance, ϵ, is a prespecified value that is dependent upon screen size and resolution. Given that the best

acceptable student performance is within ± 1 mm accuracy in positioning the graphics tablet pen cursor, for a 25.4 cm screen with 1024 pixel width, this translates into (1024/25.4) * 0.1 ~ 4 pixels.[3] This implies that any input or target vector should be at least 40 pixels in length if the inherent error is to remain within a 10% error limit. Consequently, small and highly-detailed objects should be avoided for use as target objects.[4] Note that the concept of inherent tolerance has a precedence in Thomason (1973) and Gaines (1977) who view the process of matching as one involving the solution of fuzzy sets. Fuzzy set theory, while applicable here, is not employed due to complexities of notation within the algorithms and the fact that an algebraic approach is satisfactory.

Thus two points are said to correspond if one point lies within the bounding circle of the other point as in Figure 17. However, this is approximated by a square region due to the simplicity of calculation of a point such as a as opposed to b since

$$\text{point a} = (x - \epsilon,\ y + \epsilon)$$
$$\text{whereas point b} = (x - \epsilon\cos\theta,\ y + \epsilon\sin\theta).$$

Another reason for selecting point a rather than b is that this eliminates situations such as b = (3.5,4.5) (Figure 18). As can be seen from the diagram, this point is at the

---

[3]This error is due to a combination of human coordination and random errors inherent within the electromagnetic mechanism of a graphics tablet.

[4]The combination of smallness and fineness of detail pose problems for areas not related to CAI or pattern matching as well.

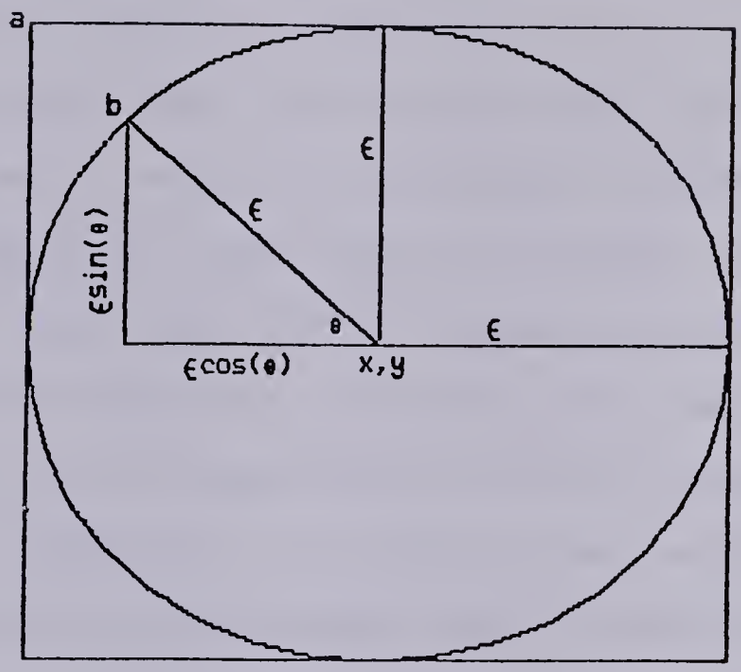


Figure 17 Point bounded by Circle and Square of Tolerance

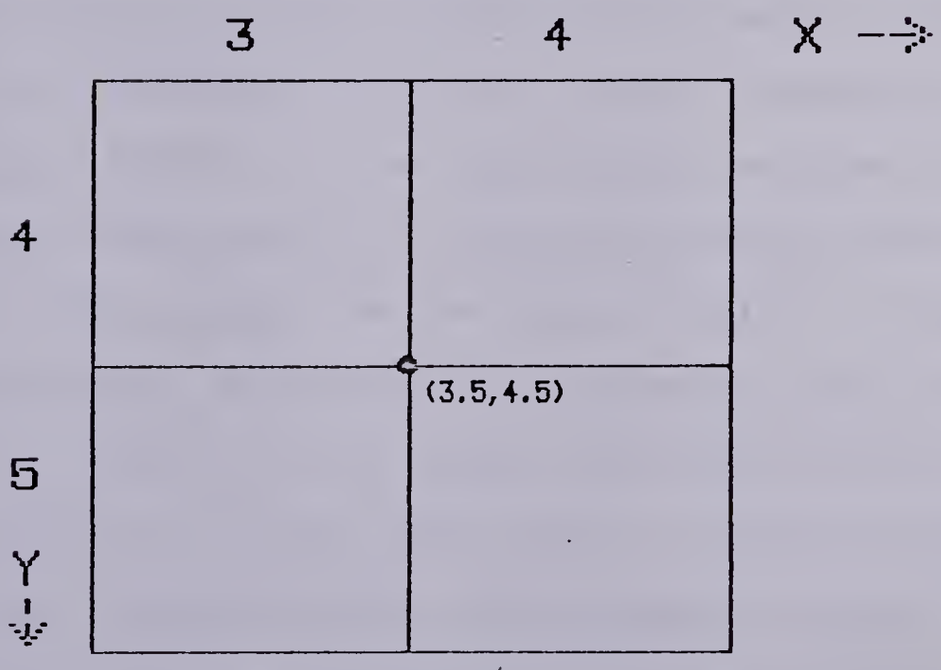


Figure 18 Four nearest Pixels of a Point

intersection of four pixels, which raises a dilemma, one which is avoided by choosing the approximate solution, of which pixel or pixels to reference.

For a vector this implies that there are bounding rectangles which establish the limits for correspondence as shown in Figure 19. Again due to increased complexity of computation, the less accurate bounding parallelogram is used instead (Figure 20). Similarly, for each of the other data types there are equivalent bounding figures that can be produced by applying $\pm \epsilon$ to each of the coordinates specified within the parameter list of the data type.

G. Level I Algorithm

As previously mentioned Level I analysis checks each primitive for an exact match (within tolerance) with an equivalent primitive from the target object. Once found, the input primitive is placed in the correct match array, **Pci**(n), and the target primitive is also placed in a correct match array, **Pct**(n). If the input primitive does not match any of the target primitives (matching parm:ip with parm:tp) and is of type circle then the input primitive is moved from {ip} into **Pwi**(n), which is the wrong match array for input primitives. This action is taken since the decision matrix clearly shows that input type circle is best matched with target type circle and that target types of point, vector, and arc are highly unlikely cases for a match with the student input of circle. As well, the corresponding target

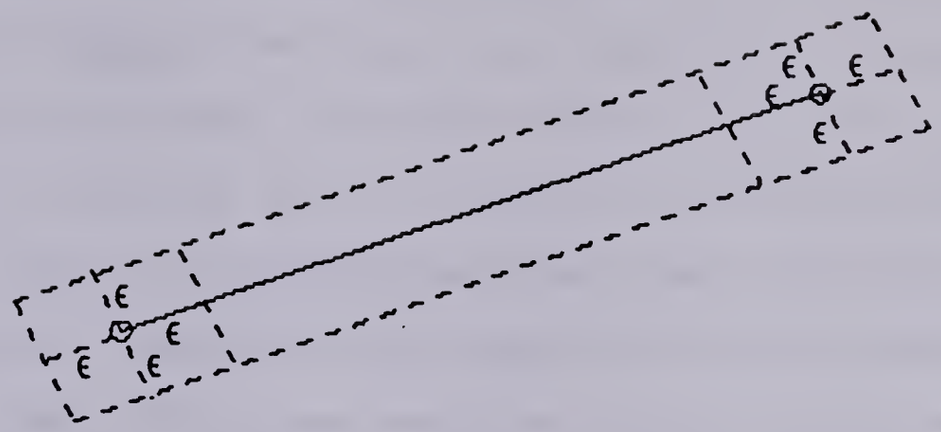


Figure 19 Exact Solution to Vector Correspondence using Bounding Rectangles

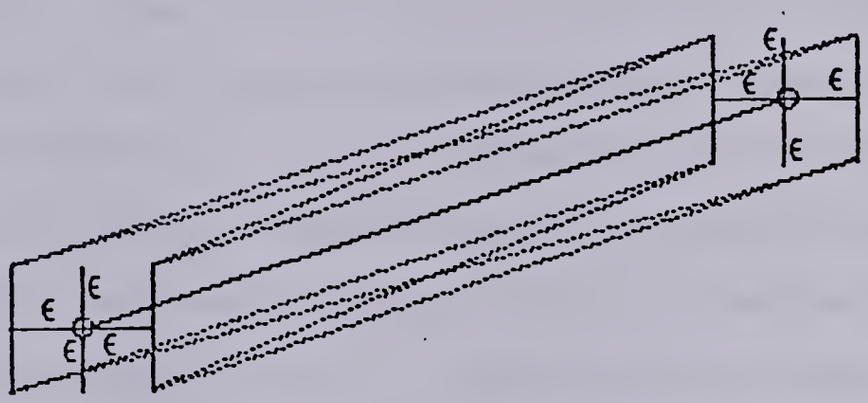


Figure 20 Approximate Solution to Vector Correspondence using Bounding Parallelograms

primitive is moved from {tp} into **Put**(n), the unmatched array for target primitives. For all other data types, whether no match or partial match (parm:ip <> parm:tp or parm:ip ~<> parm:tp), the input primitive is moved from {ip} into **Pui**(n), the unmatched array for input primitives. No movement is performed on the target primitives in this case.

The reason for moving ip into **Pui**(n) is that there is a possibility for further analysis at Level II. For example, if type:ip = vector then there is a possibility that the ip is linked to two other collinear vectors whose vector sum is equivalent to the tp. Another example is the case where type:ip = type:tp = rectangle and the input and target rectangles are specified using opposite diagonal pairs of vertices. Again, this case would be correctly analyzed at Level II. A correctly matched object (all primitives matched) causes said object to be moved into **CMi**(n), the correct match array for input objects. An object which is incorrectly matched is placed in the wrong match array, **WMi**(n) if s1 = 1 and s2 =0. If s2 = 1 then it is possible that the input object may be matched against another target object whose location in I is indicated by the pointer list. Analysis would then be repeated using the current input object and the new target object. If s1 = 0 and s2 = 0, then the remaining components of i, {ip} and **Pui**(n) must be analyzed at higher levels of the hierarchy if permitted by the decision matrix. If no further levels are permitted and #{ip} <> 0, then the input object must be placed in **WMi**(n), that is, the wrong match array.

Note that it is possible to obtain a figure of matching using the ratio #Pci(n)/#{ip} (where # indicates the number of elements), however it is difficult to assess the meaning of this ratio in terms of a "closeness" of fit (match). The above process is repeated for all input objects. If #Pui(n)

<> 0 then the analysis must continue at the next level of the hierarchy specified by the results of the decision matrix.

H. Level II Algorithm

All primitives from {ip} or **Pui**(n) of type vector are checked for collinearity (Jacobsen, 1968). At each stage of success on this check, the two vectors are summed to produce a single vector which is then checked against each vector in {tp}. Failure to find a match results in further collinearity checks with other vectors in {ip} or **Pui**(n). Failure to extend collinearity any further causes a check on the status of s2. If s2 = 1, there is a possibility that another object is actually the intended target. Analysis must begin again with the new target object. If both s1 and s2 equal zero, then this would suggest that a higher level of analysis is required and the vector is placed in **Pui**(n). If a match occurs, then the summed ip is removed from {ip} and placed in **Pci**(n), while the tp is removed from {tp} and placed in **Pct**(n).

All primitives from {ip} or **Pui**(n) of type triangle, rectangle, and polygon are decomposed into type vector. All primitives of the above type in {tp} are also decomposed into type vector. Once decomposed, analysis proceeds as in the above instance.

All primitives of type arc are decomposed from the three coordinate pair specification (start point,

"midpoint", and end point) into start point, end point, and centre of circle containing the arc (from elementary geometry). This process is repeated for {tp} as well. A match results in ip being moved from {ip} or **Pui**(n) into **Pci**(n), and tp moved from {tp} into **Pct**(n). Failure to match causes ip to be moved from {ip} or **Pui**(n) into **Pwi**(n) and tp to be moved into **Put**(n).

If the target is specified as a list of string descriptors such as "triangle", "rectangle", "equilateral triangle", and "regular pentagon", then the analysis begins with a check on the input type. If the input type, type:ip, matches and the target is nonattributed (i.e., no qualifying descriptor such as equilateral, regular or right-angled), then a correct match has been detected and the input object is placed in **CMi**(n). If type:ip = vector and does not match, then a search in {ip} is made for other vectors which can be connected to the current ip. As an example, suppose that the target descriptor is "right-angled triangle" and type:ip = vector. Since a triangle requires three vectors, a search is performed on {ip} for two other vectors whose end points can be connected in such a fashion to form a closed figure. If the search is successful, then the specifications of the attribute must be met. In this case, adjacent vectors must be checked to determine if they meet at a 90° angle (there are three vector pairs to check). If successful, the object is placed in **CMi**(n), otherwise the remaining objects in I must be checked. Failure to match results in the input

object being placed in **WMi**(n). Note that this is a table-driven process, that is, for every string descriptor permitted there must be a table entry containing attribute information such as: three closed vectors, 90° angles, length of vectors = constant, and so forth. This process does **not** function with the point data type.

If #Pui(n) + #{ip} = 0 and #Pwi(n) = 0 then the object is placed into **CMi**(n), otherwise if #Pui(n) + #{ip} = 0 and #Pwi(n) <> 0 then the object is placed into **WMi**(n). Further analysis occurs if the set of input objects has not been completely analyzed (#I <> 0) or if the #{ip} <> 0 and the decision matrix has specified additional levels of analysis.

I. Level III Algorithm

This level of analysis is designed primarily to handle input of type point. If both target and input data is of type point and the number of primitives is 100 or less, then Level I analysis should be sufficient. When the number of primitives exceeds 100, but is less than or equal to 1200, and the target data type, type:tp, is any of the seven defined types, then Level III analysis is appropriate. If the number of primitives exceeds 1200, then Level IV analysis would be appropriate. Level III analysis assumes that the target exists within a region represented by a matrix whose size does not exceed 1024 x 1024, which was assumed to be the maximum resolution of a monitor screen (1 to 1 correspondence between matrix and screen display). The

limitation on the input using 1200 primitives will be 450 x 450 (based on a regular, closed figure; actually closer to 424 x 424). Using a conventional full template approach (as opposed to the Uhr & Vossler operator matrix of Chapter 2) in which the input matrix sequentially scans across and then down the target matrix would require a check for maximum correspondence at $(1024 - 450 + 1)^2 = 330,625$ positions. At each position a maximum of 450^2 comparisons could be undertaken, yielding an approximate upper limit of 6.7×10^{10} operations. This is, of course, unacceptable for most time-sharing CAI systems. A solution is to limit the area of search (window) within the target matrix. If the target contains only one object, this can be easily accomplished by determining the smallest and largest x-coordinate and y-coordinate, which then describes the smallest bounding rectangle completely containing the target object. If the target contains more than one object, then only the current object under consideration is examined to obtain the smallest bounding rectangle. Since the target region is essentially a sparse matrix (most of the cells are zeros), the area of search can be further limited by calculating the centroid for the target object. Once found, a sequential scan using the input matrix starts with its bottom right corner positioned at the centroid (see Figure 21). The scan continues with the input matrix being shifted one position to the right until the leftmost boundary of the input matrix reaches the position of the target centroid. At this time

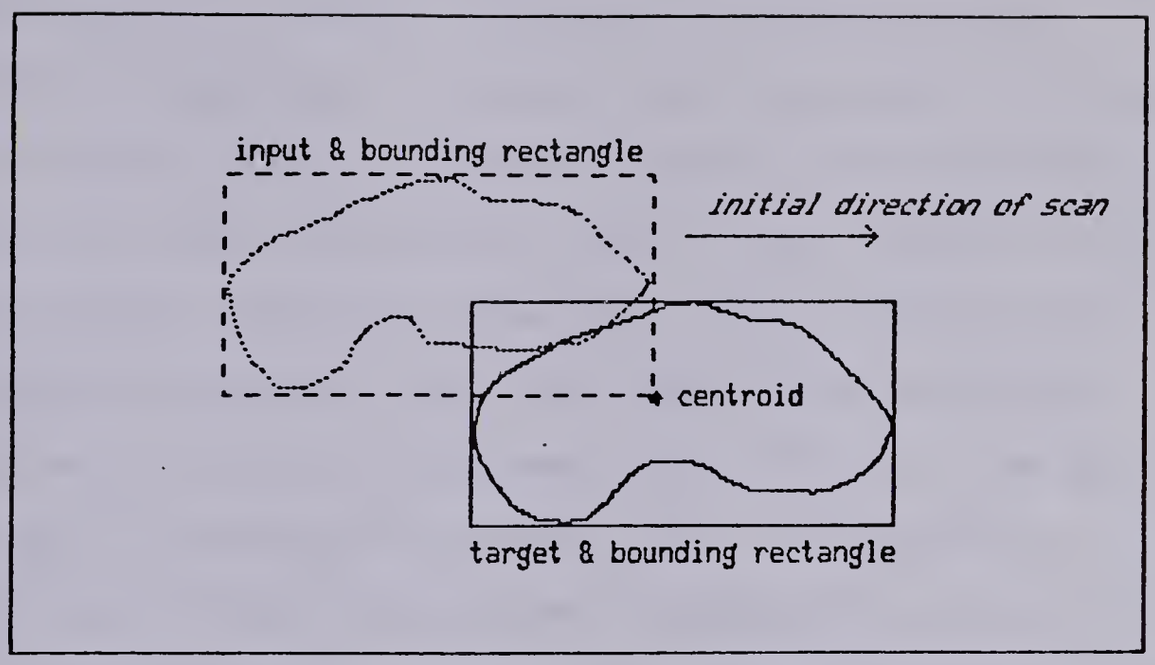


Figure 21 Sequential Scan with Input Matrix at Centroid

the input matrix is shifted one position down and starts the traversal across the target, but in the reverse direction until its rightmost boundary reaches the centroid position. The entire process is repeated until the topmost boundary of the input matrix is scanned across the target.

Barnea and Silverman (1972) used a sequential scanning technique in which they accumulated errors between input and target at each position within the search window but instead of evaluating to completion the error function at each window position, a maximum allowable error was established which when exceeded would result in termination of any further evaluation at the current window position.

The position would then be incremented and the process repeated again. A further reduction of operation was achieved by evaluating at random locations rather than at every location within the search window. Random checking within the window was possible due to scene density, that is, the images they were using had complex features with grey-scaling and hence, were not represented as sparse, binary matrices as in the present case. However, the general technique of accumulating errors until a threshold is reached is certainly applicable. An arbitrary criterion of 25% incorrect matching will be used. That is, given a figure with 800 points, closeness of correspondence will be terminated when there is a 200 point error between input and target, at which time the search window will be incremented to the next location. Further savings of computation and time can be realized if no attempt is made to maximize the correspondence and if the area of search can be limited even further. This can be accomplished by setting a lower limit for correct matching of 95% and by initiating the sequential scan with the input objects's centroid initially located at the target's centroid. If an incorrect match is found, the input's centroid is relocated at the next cell immediately to the right of the target centroid. The process is repeated, "spiralling outward" (Figure 22) from the position of the target's centroid until a match is found or until the input's centroid is 80 points away from the target's centroid (80 points (pixels) ~ 1 cm on the monitor screen)

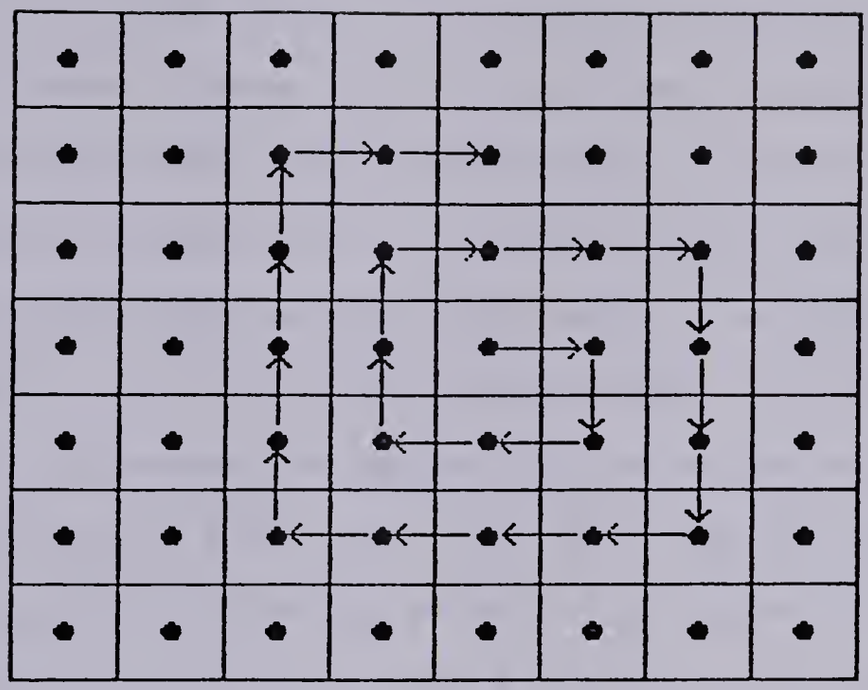


Figure 22 Spiralling Scan starting at Centroid

at which time, the process is terminated. The 95% lower limit implies that an 800 point object will be judged correct and "accurate" if no more than 40 of its points lie beyond the "template specifications". In this instance, template specifications means that point for point correspondence should be within two points, although Levels I and II utilized a four point tolerance. The choice of two point tolerance means that a 24-nearest neighbor weighting operator can be used, with the center cell at the location of the target point. At each scan position a check for correspondence between input and target will be made. At every point of the target, a weighting operator will be centered and the number of "hits" with the input object will be tallied.

Figure 23 shows a portion of an input object and a portion of a target object. At the scan location (window position) shown, there are two locations of exact matches, A and B. Assuming that the objects being discussed are closed, well-behaved figures (e.g. no singularities) and, hence, are digitally-connected, at every location in which there is an exact match, there will be found at least four more hits within the 24 nearest neighbors of the weighting operator as shown at location A of Figure 23. This implies that if each exact hit yields a minimum score of 13 (based upon operator weightings of Figure 23), then a fortiori, an object having p points must yield an aggregate score of at least 13 p. The exact aggregate can be calculated only by applying the operator against the target. For objects or primitives that are not closed, an exact match will yield a minimum score of 13p - 8 (based upon two end points). Similarly, the maximum score can be evaluated. For the majority of closed objects this will be 19 p, while for the majority of open objects it will be 19p - 14.[5] It is interesting to note that objects which yield a score closer to 13 p will have major diagonal components, while a score closer to 19 p indicates major vertical and/or horizontal components. There now exists two means to prevent overscanning by the student (i.e. the student specifies an input object which is significantly

---

[5] This does not necessarily hold for objects with cusp-like boundaries since an exact match could produce more than four auxiliary hits within the 24 nearest neighbor operator and hence, a greater aggregate score than 19 p (or 19p - 14 for the open object).

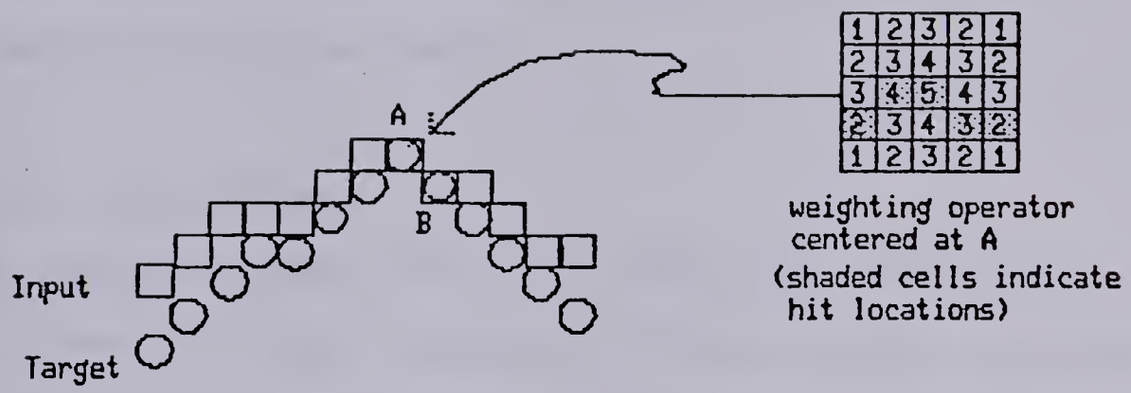


Figure 23 Weighting Operator applied at an Exact Match Point

"larger" or completely covers the target, as in a solid rectangular region, and consequently tricks the algorithm into evaluating a correct match). The first is to ensure that the number of input points does not exceed p by more than, say 20 per cent, and that the aggregate score does not greatly exceed 19 p. With regards to no matching at the tested point or pixel of the target, an arbitrary score of -16 (based upon the 13 and 19 values above) will be applied for each error location and will be continued until 25% error occurs at which time the input matrix will be shifted to the next test location. A score between 75% and 95% indicates that while the match is good, there exists sufficient anomalies to reject the exact match

classification. Appropriate error messages can be generated to inform the student of this fact. As in the cases of Level I and Level II analyses, switches s1 and s2 are checked to determine if there are alternative likely targets. If not, the input object is described as being incorrect and is subsequently moved into **WMi**(n).

J. Level IV Algorithm

As reported under Level III Algorithm, Level IV is employed when the input exceeds 1200 data points. Although this limit is rather arbitrary, it prevents potential bottlenecks due to excessive processing using a sequential scan approach. Level IV analysis is used to accommodate objects consisting of much more than 1200 points, but which are still confined to a window of 1024 x 1024 (limits of hypothetical monitor screen). Once again the assumptions of representation of target data in a matrix form are applicable.

The Level IV analysis is based on the pioneering work of Hu (cited in Hall, 1979, pp. 420-423) and Alt (1962) using moment invariants. Generally speaking the process is as follows.

Given a function of two variables, f(x,y) which represents an image function, the moment of order j+k is defined as

$$m_{jk} = \int \int x^j y^k f(x,y) dxdy$$

For a binary, digital image whose cells (pixels) are of unit area this becomes

$$m_{jk} = \sum_{x}\sum_{y} x^j y^k \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Taking this with respect to the centroid yields the central moments

$$\mu_{jk} = \sum_{x}\sum_{y} (x - \bar{x})^j \, (y - \bar{y})^k,$$

where $\bar{x} = \dfrac{m_{10}}{m_{00}}$ and $\bar{y} = \dfrac{m_{01}}{m_{00}}$.

In this instance, m_{00} = area of the image (number of cells constituting the image since each cell has unit area), and

$$m_{10} = \sum_{x}\sum_{y} x \quad , \text{ and}$$

$$m_{01} = \sum_{x}\sum_{y} y.$$

Thus $\mu_{00} = m_{00}$

$$\mu_{10} = \sum_{x}\sum_{y} (x - \bar{x}) = \sum_{x}\sum_{y} x - \sum_{x}\sum_{y} \bar{x}$$

$$= m_{10} - \frac{m_{10}}{m_{00}} m_{00}$$

$$= 0$$

Similarly,

$$\mu_{01} = 0.$$

$$\mu_{20} = \sum_{x}\sum_{y} (x - \bar{x})^2 \, (y - \bar{y})^0 = \sum_{x}\sum_{y} (x^2 - 2x\bar{x} + \bar{x}^2)$$

$$= m_{20} - 2m_{10}\frac{m_{10}}{m_{00}} + \frac{m_{10}^2}{m_{00}^2} m_{00}$$

$$= m_{20} - \frac{m_{10}^2}{m_{00}}$$

Similarly,

$$\mu_{02} = m_{02} - \frac{m_{01}^2}{m_{00}}$$

and the remaining moments up to and including order 3 are

$$\mu_{11} = \sum_{x}\sum_{y}(x - \bar{x})^1 (y - \bar{y})^1 = m_{11} - \frac{m_{10}m_{01}}{m_{00}},^6$$

$$\mu_{30} = \sum_{x}\sum_{y}(x - \bar{x})^3 (y - \bar{y})^0 = m_{30} - 3\bar{x}m_{20} + 2\bar{x}^2 m_{10},$$

$$\mu_{12} = \sum_{x}\sum_{y}(x - \bar{x})^1 (y - \bar{y})^2$$

$$= m_{12} - 2\bar{y}m_{11} - \bar{x}m_{02} + 2\bar{y}^2 m_{10},$$

$$\mu_{21} = \sum_{x}\sum_{y}(x - \bar{x})^2 (y - \bar{y})^1$$

$$= m_{21} - 2\bar{x}m_{11} - \bar{y}m_{20} + 2\bar{x}^2 m_{01}, \text{ and}$$

$$\mu_{03} = \sum_{x}\sum_{y}(x - \bar{x})^0 (y - \bar{y})^3 = m_{03} - 3\bar{y}m_{02} - 2\bar{y}^2 m_{01},$$

From these, Hu has shown that a set of seven invariant moments may be derived by first applying the following normalization.

$$n_{jk} = \mu_{jk} / \mu_{00}^{\gamma}$$

where $\gamma = 1/2(j + k) + 1$

for $j + k = 2,3,....$

For the purposes of GMATCH, only the first two invariant moments are considered since the remaining five (which involve higher-ordered terms) appear to be more sensitive to noise and show greater variations in the results of empirical studies (Alt; Hall, 1979, p. 423). The first two invariant moments (invariant to translation, rotation and scale) are

$$\emptyset_1 = n_{20} + n_{02}$$

---

[6] Note the similarity of these equations to those of the mean, variance, and covariance up to this point.

$$\emptyset_2 = (n_{20} - n_{02})^2 + 4n_{11}^2$$

Typically, one should expect deviations in $\emptyset_1$, within 0.2 and in $\emptyset_2$ within 0.5 for images which are similar (i.e. have undergone any of the affine transformations except mirroring).

To establish congruency it becomes necessary to examine the non-normalized, non-referenced moments generated by equation (1) up to and including order six. However, in practice one would expect that two objects are congruent if the moments of orders one, two, and three are equal. If an object or set of primitives show deviations between target and input greater than that specified above, then there exists sufficient grounds to reject claims of similarity and hence, move the object into **WMi**(n). Objects which are not judged to be correct at this level, must necessarily be judged as incorrect or unresolved since no further analysis from Level IV is possible at this time.

K. Level V Algorithm

Level V analysis is, generally speaking, the last resort for matching after failure at Level II. Level V analysis is further restricted to closed figures, that is, polygons. One of the earliest works in this area is that of Lee (1974), who proposed a shape-oriented dissimilarity measure based upon the measure of the interior angles of triangles and polygons. This approach assumed that the

objects being compared had equivalent numbers of vertices and, hence, angles. More recently, Kashyap and Oommen (1982) proposed a technique, using either angle measures or edge measures, which is not restricted by the numbers of vertices in either the target or input. The approach used here is essentially the edge-based algorithm of Kashyap and Oommen. It proceeds as follows.

Both input and target are normalized to unit perimeter and rotated so that edge i of the input and edge j of the target are in a horizontal position. The required transformations are then saved. Next, the centroids are computed for both figures. Assuming m edges for the input and n edges for the target, there are m x n possible orientations of both figures with respect to each other's edges. That is, the figures are oriented such that the ith edge of the input coincides with the jth edge of the target at their midpoints. The centroid to centroid distance, $\delta^{i,j}$, in this orientation is now computed. This measure is calculated for all possible m x n orientations of edges. This set of $\delta^{i,j}$'s is placed in ascending order and the first m + n values of (i,j) which yield the lowest δ's are kept. This set of lowest values of $\delta^{i,j}$ is referred to as Γ,

$$\Gamma = [(i,j) \mid (i,j) \in (1,\ldots,M) \times (1,\ldots,N),\ \delta^{i,j} < \delta^*]$$

where δ^* is the threshold value just greater than the (m + n)th value of δ.

For each of the (i,j) pairs in Γ, the integral square error between figures is calculated by first taking the

normalized, edge-oriented figures and establishing a new coordinate system with the origin located at the midpoints of edges i and j, and the abscissa lying on those edges. Traversing from the origin in a clockwise direction on both figures simultaneously yields a point $H_i(\lambda)$ on the input figure and a point $G_j(\lambda)$ on the target figure after a distance of λ. If the distance, λ, is less than or equal to the distance from the midpoint of the edge to its end point, then the coordinates of $H_i(\lambda)$ will be $(\bar{x} - \lambda, 0)$, where $\bar{x}$ is the x-coordinate of the midpoint. However, if a movement of $\lambda - 1$ reaches the end point of the edge, then the remainder of the distance, 1, must be traversed along the adjacent edge. By application of ratio and proportion it is seen that the coordinates of $H_i(\lambda)$ will be

$$\left(x_1 + \frac{1}{d}\Delta x,\ y_1 + \frac{1}{d}\Delta y\right)$$

as shown in Figure 24.

Now the dissimilarity between input and target after a distance of λ can be characterized by the integral square error criterion,

$$||H_i(\lambda) - G_j(\lambda)||^2$$

Evaluating this along the complete perimeter yields a total dissimilarity

$$D_{i,j} = \int_0^1 ||H_i(\lambda) - G_j(\lambda)||^2 d\lambda.$$

which can be approximated by

$$\sum_{k=0}^{K-1} || H_i(\lambda_k) - G_j(\lambda_k) ||^2 \delta$$

where $K\delta = 1$. K is the number of points at which $H_i(\lambda)$ and $G_j(\lambda)$ are evaluated (in practice, K = 20 should prove to be

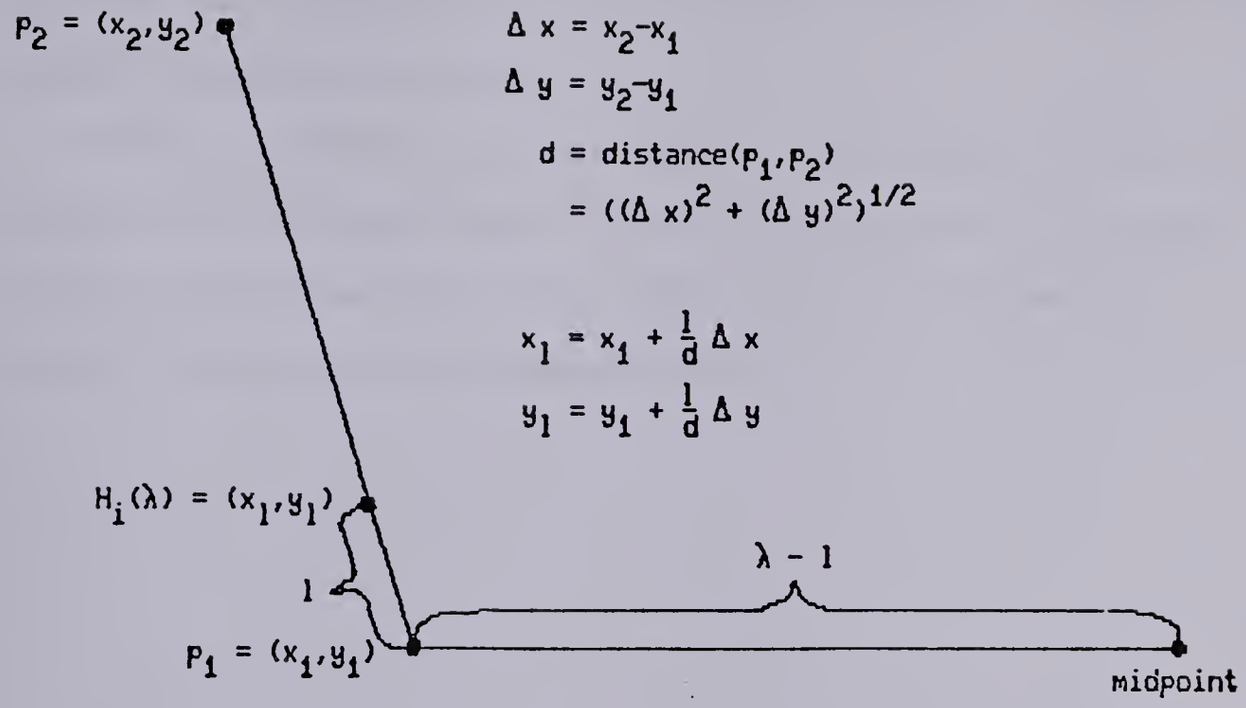


Figure 24 Calculation of $H_i(\lambda)$, where λ exceeds the Current Edge

sufficient for most figures encountered). Now $D_{i,j}$ is evaluated for the m + n values (Γ) of (i,j), but only the minimum value is of interest since it occurs for that orientation which yields the greatest degree of similarity or conversely, the least dissimilarity.

Therefore, $\bar{D}(\text{input,target}) = \min_{(i,j)\epsilon\Gamma}[D_{i,j}]$

In order to reduce the total computational effort, the current computation of $D_{i,j}$ is stopped as soon as the computation exceeds the previous low value of $D_{i,j}$. If the current value of $D_{i,j}$ is lower than the previous one, it displaces the previous value. This process continues until all m + n cases are examined. Kashyap and Oommen discovered

that this reduced computational time significantly, having computed (to completion) less than one percent of the total number of integrals required.

This procedure not only determines whether or not two figures are matched (input moved into CMi(n)), but also yields a pseudometric which indicates to a student or teacher the degree of dissimilarity.

IV. TESTS, RESULTS AND DISCUSSION

A. Behaviour of GMATCH under various inputs

To test the efficacy of GMATCH, it was decided that two general types of input be used. One type would correspond exactly with the target sets with respect to number of primitives, type of primitives, and the specific parameter list. This type of input is hereafter referred to as sympathetic data. The second type, referred to as nonsympathetic data, may differ with respect to the number, the type, and the parameter list, but remains compatible (compatibility was defined in Chapter Three as capable of being resolved by the decision matrix) to the target set. The reason for testing nonsympathetic sets is that a learner using GMATCH, within the context of some CAI lesson, may respond in a manner not anticipated by the author. That is, the student uses not only different data types, but also a different sequence of construction and number of data types. (A particular example of this might be a geometry lesson which requires a student to produce a hexagon inscribed in a circle using typical geometric construction procedures. The probability that both student and author use exactly the same steps in the construction is extremely remote.) Hence, it is necessary to determine if the response of GMATCH in analyzing such sets is acceptable, since one of the claims of GMATCH is its ability to handle diverse input structures in a relatively parsimonious way. In this case, "acceptable"

means that the response of GMATCH corresponds to the decisions made a priori by the researcher on the basis of visual comparison, and that the response times are within a designated range of 0 to 120 seconds. The 120 second upper bound is an arbitrary limit chosen by the researcher, beyond which currently available improvements in software and hardware are not likely to yield response times that are desirable and typical of existing CAI systems. Times on such typical systems tend to be within the 10 second range.

It was anticipated that the results of any tests involving Levels I and II of the hierarchy would indicate a linear or nearly linear relationship between the number of primitives in the input and the CPU time for execution of the GMATCH procedure, since these levels are functionally linear in operation. On the other hand, Level III analysis is algorithmically related to the square of the smallest dimension of the bounding rectangle containing the input. Thus the results for the test of Level III should indicate a quadratic relationship between the number of primitives and CPU time.

B. Use of GMATCH in a CAI environment

Although the analysis routines and decision matrix constitute the major portion of GMATCH, the interface components are equally important. For an author this means being able to specify, create, or modify appropriate target objects for a particular CAI lesson in an uncomplicated a

manner as possible. As well, the creation of code which is used to control the GMATCH operation must be easily learned. For students this also implies that its use must be easily learned and remembered. There are at least three measurements which can be undertaken to provide data concerning the usability of the GMATCH interfaces:

1. the measurement of the time it requires to train either authors or students,
2. the measurement of the latency time (before responding to some arbitrarily chosen task) as a function of training time, and
3. subjective evaluations provided by the learners, in this case, both authors and students.

If the GMATCH interfaces are well-designed in the human factors sense, then it would be expected that:

1. the training time is relatively short (on the order of 10-15 minutes),
2. increased training either decreases the latency time or the latency time remains constant (implying that minimum training already produces the optimal response times), and
3. subjective evaluations by both authors and students would be highly favorable.

Additional measurements and studies could also be undertaken, but these would most likely involve a particular application of GMATCH within some curriculum.

C. Tests performed and not performed

Since Levels IV and V have already been sufficiently tested and documented in the literature, it was decided that only Levels I, II, and III would be programmed and tested. Furthermore any tests of Level III would be impacted by the fact that only a software emulation of the algorithm described in Chapter Three could be performed. The original algorithm requires the capabilities of reading screen memory, and bit block transfers of screen RAM directly to other RAM areas. Since these capabilities were not available with the test equipment at hand, this precluded any capacity for the decision matrix to accommodate analysis of input of type point against the other data types for targets. For instance, a target object specified by the arc data type and an input specified by the point data type cannot be analyzed unless there is an ability to read the video RAM so that a binary matrix can be extracted for the arc data type. This matrix can then be compared with that matrix generated by the point data type. Another related degradation of Level III was that the CPU times for this routine would be at least an order of magnitude greater than what is possible with the requisite hardware. These shortcomings of the implemented version of the Level III analysis would not affect its potential unless the reported CPU times exceeded the typical times by at least two orders of magnitude.

With respect to any testing of the ease of use of the interfaces, it was a moot point since there was no existing

CAI environment in which to embed the GMATCH environment. Consequently, any interface constructed for these tests would not be representative of the interfaces as they were described in Chapter Three. It was decided that no meaningful tests could be conducted at this time.

D. Results

A subset of the GMATCH environment described in Chapter Three was developed, coded and tested on a Digital Equipment Corporation (DEC) Vax 11/780 minicomputer using Vax VMS operating system Version 3.4 at the Division of Educational Research Services, University of Alberta, Edmonton. Additional hardware included a DEC GIGI terminal, Barco RGB monitor, and a Summagraphics Bit Pad One digitizing tablet with pen cursor. The GMATCH subset included Levels I, II (omitting the arc and string components), and III of the analysis hierarchy and a combination author/user test interface; all of which were coded in Vax-11 BASIC Version 2.1. The choice of BASIC over the other available languages such as FORTRAN or Pascal was predicated by the following:

1. the availability of intrinsic features and commands not directly available with the other languages, such as dynamic string allocation and matrix operators, and
2. taking advantage of procedures previously developed by the researcher for the program DRAW2.0 (Lee, 1983).

Minor differences in the format of the data structures (e.g. 4,193,300,80,217 specifies a rectangle with opposing

vertices at coordinates 193,300 and at 80,217) did not affect either the operation of the hierarchy or the generality of the results. Also, the implementation and operation of the decision matrix differs from that described in Chapter Three due to the reduced number of algorithms tested, but remains faithful to the concept of differentiating analysis based upon variation in data types.

Five sets of target data based upon the primitive types point, vector, circle, and rectangle (see Appendix C) were constructed for testing of the GMATCH hierarchy subset. Each set, with the exception of points, was further subdivided into subsets consisting of 1, 2, 5, 10, and 20 primitives. These sets were Set I (vectors), Set II (rectangles), Set III (circles), Set IV (combinations), and Set V (points). Set V consisted of three subsets, one with 87 points, one with 102 points, and another with 117 points. (These sets were necessarily kept small since the data points were hand-coded rather than entered via the digitizer.) Sets I to IV were tested using input sets that were sympathetic and nonsympathetic. The nonsympathetic tests were conducted on the subsets consisting of 5, 10, and 20 primitives. For all of these cases, the number of input primitives was equal to the number of target primitives. Tables 1 and 2 show the results of the sympathetic and nonsympathetic tests. An additional test was performed on the 5-vector subsets of Sets I, II, and III using nonsympathetic input sets of 5, 10, 15, and 20 primitives. The results of this test are

shown in Table 3. Table 4 shows the results of the Set V tests which are plotted as Figure 25. All results are shown as CPU times (to the nearest hundredth of a second).

Table 1 GMATCH Response Times in Seconds for Sympathetic Input Sets

	Number of target primitives				
	1	2	5	10	20
Set I (vectors)	0.08	0.08	0.15	0.30	0.55
Set II (rectangles)	0.07	0.09	0.14	0.23	0.49
Set III (circles)	0.06	0.08	0.11	0.18	0.43
Set IV (combinations)	/	/	0.14	0.25	0.48

Table 2 GMATCH Response Times in Seconds for Nonsympathetic Input Sets

	Number of target primitives		
	5	10	20
Set I (vectors)	0.17	0.30	0.72
Set II (rectangles)	1.18	2.99	9.60
Set III (circles)	0.12	0.21	0.45
Set IV (combinations)	0.42	0.85	2.42

Table 3 GMATCH Response Times in Seconds for Five Vector Subset

	Number of input primitives			
	5	10	15	20
Set I (5 vectors)	0.17	0.3	0.39	0.52
Set II (5 rectangles)	0.17	0.31	0.52	0.65
Set III (5 circles)	0.11	0.15	0.18	0.20

Table 4 GMATCH Response Times in Seconds for Level III Algorithm

Number of target points	Number of cycles through algorithm[7]					
	1	5	10	15	20	25
87	.10	3.67	10.78	21.59	35.98	54.27
102	.10	4.21	12.45	24.97	41.73	62.39
117	.35	4.92	14.80	29.79	49.17	73.46

---

[7] Each cycle through the algorithm moves the input matrix an equivalent number of points away from the location of the target centroid.

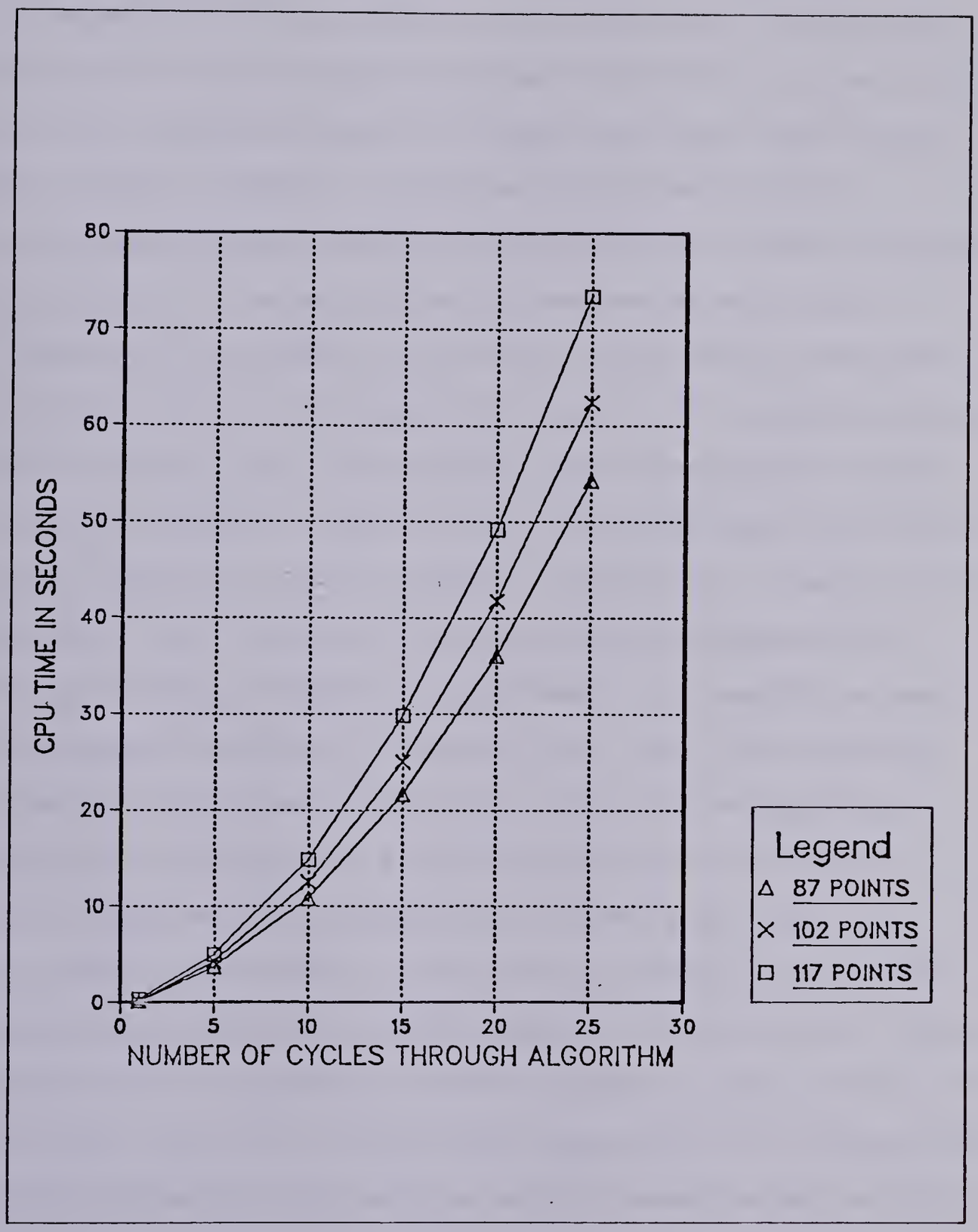


Figure 25 Graph of GMATCH Response Times for Level III Algorithm

E. Discussion

In all cases, the decisions made by the algorithms corresponded to those made by the researcher (before the execution of the analysis routines) concerning correct and incorrect matches between the input sets and target sets. The results of Table 1 indicate a relatively linear relationship between execution time and the number of target primitives in the subset, which was predicted by the linearity of the Level I algorithm. Furthermore, with the exception of Set II (rectangles) and Set IV (combinations), this linearity also extends into the nonsympathetic test results of Table 2. Again, this is as predicted since both Level I and Level II are linear in operation. However, both the Set II and the Set IV results of the nonsympathetic tests of Table 2 indicate significant nonlinearity as well as a marked increase in computational time. This apparent anomaly is resolved by the fact that the nonsympathetic tests of rectangles used the vector primitive as input. Thus, considerable manipulation of the target data is involved in both Level I and Level II analysis since the target rectangles must be decomposed into equivalent vector primitives and reordered before analysis occurs in Level II. In fact, the test is pictorially equivalent to a comparison of 80 unordered input vectors to 80 ordered target vectors. But, the results are not computationally or algorithmically equivalent to the 80 vector case due to processing at both Levels I and II. Since Set IV also contains a number of

rectangles, a similar marked increase in execution time and nonlinearity is seen. It should be noted that changes in the order of the input primitives will also significantly affect execution times for the larger subsets.

The results of Table 3 also indicate an almost linear relationship between the number of input primitives and the number of target primitives. Again, this is in agreement with expectation due to linearity of algorithms. Although nonsympathetic input sets were used for the Table 3 results, the computational times are lower due to the use of the same primitive types as in the target sets. Consequently, there is also a decreased number of combinations of primitives to examine. Hence, the expected nonlinearity for Set II that was evident in Table 2 does not appear in Table 3.

An examination of Table 4 and Figure 25 indicates that the cycle times through the Level III algorithm display approximately quadratic behavior. This is in agreement with the predicted behavior since the number of calculations performed increases proportionally to n(n-1) as the digital distance to the target centroid (n) is increased (i.e. the size of the search area becomes increasingly larger). In the event that the appropriate hardware is not available, and that Level III must be implemented in software, it would seem appropriate to limit the maximum number of cycles through the algorithm to 25 rather than the 80 cycles mentioned in Chapter Three. This is based upon the assumption that a ten-fold increase in performance (i.e.

reduction of CPU time to the seven second range) is realizable through optimization of computer code.

In general these timings compare very favorably with the results of response times for approximate string matching algorithms currently in use on existing CAI facilities.

V. Summary and Implications

A. Extensions to GMATCH

The purpose of the present study was to investigate and design a CAI tool which would permit a student to create and input a graphics image as a response to a CAI query. To be useful, such a tool would require an analysis component capable of resolving the nature (the features) of the input image. A subsequent in-depth examination of the current research in pattern recognition and image processing revealed a multitude of recognition algorithms and procedures; none of which was singly capable of resolving all recognition tasks. A decision was made to create an addressable (in the computer sense) hierarchy consisting of five levels of graphics analysis routines which were capable of resolving specific matching tasks. Furthermore, to increase the efficiency of matching, the levels of the hierarchy would be constrained by the nature of the target data (and input data) so as to create a relatively parsimonious structure. Although the initial results of matching and execution times are encouraging, it is obvious that the current design and implementation can be improved.

As previously described in Chapter Three, the current specifications of GMATCH enable the analysis of data types point, vector, arc, circle, triangle, rectangle, and polygon. This makes GMATCH suitable for use in CAI courses which involve two-dimensional, static coordinates. For

example, GMATCH can currently be used for elementary geometry, vector geometry, vector physics, elementary set theory, some aspects of motion geometry, and basic electricity. To extend its usefulness to other disciplines or curricula requires a number of obvious extensions. In this section, consideration is given to the possible accommodation for dynamic processes, or processes in which time and space interact, three-dimensional data, and character recognition as well as to improvements in general performance.

1. **Dynamic processes.** In a recent article, Olson (1981) has re-introduced (in an educational sense) the notion of the use of the medial axis function (MAF) as a means to describe and model dynamic growth processes. Such processes are often awkward to describe verbally or sufficiently complex in mathematical notation so as to blur any obvious connotation; a case in point being the growth of bread dough (described in Olson, pp. 364-365). Yet it seems reasonable to think that a mathematics curriculum could include a study of the geometry of such processes especially in a CAI environment. (The advantages of a computer to create, manipulate, and provide pertinent interaction experiences for such a study cannot be replicated by books, non-interactive video, or the classroom instructor.)

   The use of image or picture descriptions taken at various stages of the growth would seem to be a natural

means for expressing such processes using GMATCH. Recall that the MAF was rejected in Chapter Two as a means of figure encoding due to its extreme susceptibility to noise, and thus, comparing two figures on the basis of their MAF's would be difficult. (In practice, the difficulties in using the MAF are quite severe due to problems associated with generating the discrete representation from a continuous one; but see Wakayama (1982) for a new algorithm.) However, given the inverse transform, the MAF and its time-based behaviour it should be possible to reconstruct the boundaries of the growing object (process) at any desired time t_i. It then would be a relatively simple matter to generate the boundaries at times t_{i-k} and also at t_{i+k} (k is a tolerance level) and check to see whether the student's input image is within the confines of these boundaries (viz., within the tolerances).

2. **Three-dimensional data.** The studies of astronomy, Newtonian physics, molecular chemistry, biochemistry, architecture, kinesiology and many others require the use of three-dimensional data. Some specific examples are the study of planetary orbits, construction details for buildings, and bonding of atomic and molecular structures. Unfortunately, there are at least two problems associated with the handling of three-dimensional data. The first is the method of input; the second is the representation and analysis of

the data. In general the input of three- dimensional data is not readily handled with the exception of special devices such as the MCS Space Tablet (Micro Control Systems, 1982). Even this particular device has its weakness, since it functions best only when tracing a solid three-dimensional object. Its use without such solid templates is awkward. Barring the availability of such a device, an acceptable alternative (admittedly, a highly subjective point of view) is to permit input of data using a plane-slice approach. That is, two-dimensional (say, x-y) slices of the object are taken at discrete intervals of z until the entire object is "sliced". For objects displaying significant regularity this approach is not too objectionable since the z-intervals could be quite gross. However, it is unsatisfactory for those objects which display highly irregular surfaces. The obvious approach, yet an extremely tedious one, is to key in the data points one at a time (suffice to say that this is also quite untenable for relatively large convoluted surfaces). Thus, the computer as an image generator and control device for three-dimensional figures in mathematics instruction, though possible, awaits further technical advances for its usability.

With respect to the data representation and analysis thereof, some recent studies are of significance. If a wireframe approach is used, then

rather simple analytic geometry can be applied (a three-dimensional extension of Levels I and II). The problem associated with wireframe representations is that of ambiguity with respect to depth and surfaces implied, although Markowsky and Wesley (1980) have done work in eliminating the surface ambiguity (see Requicha and Voelcker (1982) for an example of this ambiguity) by producing an algorithm which generates all possible solids represented by a valid wireframe. Sugihara (1982) has devised a means of discriminating correct and incorrect line representations of polyhedrons and to correct (if correctable) inconsistent representations. Tsai (1983) has produced two algorithms for image matching and three-dimensional surface reconstruction based on multiple perspective views. Using Freeman chain codes, Wallace, Mitchell and Fukunaga (1981) have devised a means for three-dimensional analysis based on extraction of local features. Yau and Srihari (1983) have devised a tree structure for multidimensional images based on recursive subdivision of d-dimensional space into 2^d hyperoctants (analogous to the quadtree of two-dimensional space, which was discussed in Chapter Two). Other representations such as constructive solid geometry trees are discussed in a concise article by Brown (1981). Thus, a multitude of algorithms can be employed for three-dimensional work. At present it appears that no single algorithm for three-dimensional

work can satisfy all possible applications. The question of which algorithm(s) to employ seems to be deeply rooted in the particular application. A caveat is in order here. Generally speaking, most three-dimensional algorithms show exponential complexity -- that is, the amount of computation will vary exponentially with the size of the data structure. This can easily overload a small or medium-sized computer, leading to what is called a CPU-bound state.

3. **Character (script) recognition.** In many instances a student may need to point out or identify certain components of an image. As an example, consider an exercise in elementary statistics which requires the student to draw the shape of distributions which are positively skewed, negatively skewed, and symmetrical. This task requires not only graphics input, but also character input which is associated with the appropriate curve. A CAI system capable of resolving this problem requires a text recognition algorithm. There are at least two components to consider in any text recognition scheme. The first is to make sense of the character itself, and the second is to identify pertinent strings of such characters (i.e. the words). With respect to the former, Loy and Landau (1982) have described a real-time, on-line procedure based on Freeman coding and character segmentation into polygonal structures (i.e. modeled as straight line segments). Another approach

which has received a great deal of attention is the Viterbi algorithm. Tanaka, Hirakawa, and Kaneku (1982) report that a combination of modified trellis and Viterbi algorithms was used successfully to recognize handwritten English and Japanese characters.

With respect to word recognition, taking into account possible misspellings, three recent studies are of interest. Hull and Srihari (1982) compared the results of the binary n-gram and Viterbi algorithms using dictionaries as large as 90,235 words. Hull, Srihari, and Choudhari (1983) compared the hybrid-integrated dictionary Viterbi algorithm to a hybrid-cascaded approach known as the predictor-correlator algorithm. Their results confirmed the superiority of the integrated method which achieved a higher correction rate within a shorter execution time. A very promising approach based on Bayes' decision theory is reported by Bozinovic and Srihari (1982), who demonstrate the superiority of their approach over the edit-distance algorithm of Kashyap and Oommen. Unfortunately, the most telling aspect of these works is the amount of computational time and memory requirements to perform even the most perfunctory of analyses. For the time being, it is probably prudent to conclude that spelling correction and character recognition algorithms are insufficiently developed in the sense of real-time performance to include within the framework of GMATCH,

although character recognition performance on its own appears to be adequately fast.

4. **Next generation.** Of the five levels of analyses reported in Chapter Three, the third level using sequential-scanning appears to be the weakest with respect to execution time. Other techniques such as Freeman coding or improved fast Fourier transforms should be considered for future implementation. Unfortunately, there does not exist (to this researcher's knowledge) any substantial body of empirical data comparing the various algorithms when the objects to be compared are sparse and binary in nature. Hence, there is a need for thorough testing of various algorithms using a variety of images. A significant increase in performance could also be realized by the use of specialized hardware (optical character recognition devices, DMA, etc.), but at a cost of decreased flexibility in assimilating new algorithms into the overall structure of GMATCH.

In the light of a recent review of work on intelligent tutoring systems (Sleeman & Brown, 1982), it is anticipated that adaptive, multi-modal input (AMI) mechanisms will be the norm of future CAI systems. An AMI would take into consideration various physical and psychological attributes of a user and thus, tailor the input mechanisms and analysis routines to accommodate any unusual behaviours exhibited by the user. For

example, an AMI should provide for text input via keyboard or hand-drawn using a graphics tablet and also voice input and output. The adaptive mechanism would provide for essential filtering and alteration of AMI components (certain cases might require some human intervention) such that visual components could be disregarded in the case of blind individuals, voice input and output could be altered or eliminated to recognize speech impaired or prevented by certain physical disorders, dyslexic-type impairments would be recognized and accommodated, and so forth. In many respects, the availability of such mechanisms in CAI could prove to be the "great equalizer" in allowing individuals who are otherwise handicapped to function competitively in an increasingly physically-oriented environment. Of course, the creation of such mechanisms would be heavily dependent upon increased research in the areas of human factors studies, human-machine interfaces as well as improved hardware and software.

B. Pedagogical implications

Thus far little has been said about the possible pedagogical implications in using a GMATCH-based CAI system. In this section some prognostications will be made as to the possible influence GMATCH could have on the design of instruction, particularly with respect to mathematics, but not exclusively so.

Although the history of formal mathematics education extends back thousands of years, attempts to determine the nature of the learning mechanisms in humans have provided very little data to date. However, some current studies are beginning to shed some light about possible mathematics knowledge building processes. The importance of such studies is that they ultimately point the way to the improvement of instruction (teaching) and curriculum. As a guide, Carpenter and Osborne (1975) have identified three factors which they consider crucial to future research. These are a complete analysis of (a) the entering cognitive skills of each subject, (b) the specific learning effects for subjects at different cognitive levels, and (c) the specific variables of the training situation that account for learning. It would seem that the use of GMATCH, as a research tool, could extend our knowledge of the acquisition of mathematical skills, the development of concepts and the improvement of instructional strategies.

For example, many studies on learning mechanisms often use a one-on-one approach. That is, the researcher conducts individual assessments with each subject, attempting to tease out subtle manifestations of mathematical development. Although the researcher may attempt to conduct such assessments in a repetitious and unbiased manner as possible, a number of confounding factors can be introduced. These factors may involve unrecognized but overt gestures and verbalizations which subsequently influence the child's

response. On the other hand, a tool such as GMATCH can be effectively used (in a completely replicable, consistent manner that is free of in-built biases) in research domains concerning imagery and its importance in the acquisition of higher mathematical (and mental) processes.

Another research question that can be addressed by GMATCH is whether or not the restriction of the mode of input (response) will affect the subsequent cognitive development or assimilation of concepts or skills by a student? There does not seem to be any substantial body of data either supporting or refuting such a conjecture except for the obvious (viz., you cannot expect a deaf person to listen to and respond to questions from a tape recording). However, Carpenter and Osborne (1975) state that "some children understand certain principles but are unable to verbalize them. Significantly earlier development of fundamental operations has been found with nonverbal assessment techniques compared to verbal techniques" (p. 93). Again, since GMATCH provides a capability for image creation, manipulation, analysis, and record-keeping, it should be feasible to construct experiments using GMATCH to verify early acquisition of mathematical operations.

Furthermore, one might ask whether it is possible for GMATCH to mediate learning behaviour and if so, how? Based upon the research of Skemp (1975), one would be inclined to answer in the affirmative that GMATCH can mediate mathematical development. Skemp views the use of graphics

symbols as a means to reduce cognitive strain (p. 84) and any disadvantages associated with such use are "outweighed by the conciseness and clarity of the visual symbols" (p. 101). He further suggests that:

> If we are right in thinking that visual imagery is most favourable to the integration of ideas;... then we might reasonably hypothesize that persons who have been noteworthy for their contributions to mathematical and scientific understanding will be found to use visual rather than auditory imagery (p. 112);

and goes on to cite instances of such occurrences by Galton, Einstein, Kekule, and Bragg. Additional evidence is supplied by Cook and Kersh (1980), citing the studies of Fennema and Sherman (1977, 1978), Armstrong (1980), and Schonberger (1976), who state that performance on spatial tasks is significantly correlated with mathematics avoidance and with problem solving. Although it has previously been implied, it should be explicitly stated that a fundamental strength of GMATCH is that it addresses the single, most dominant sensation of humans -- namely, vision. But unlike other visually-oriented teaching tools, it encourages interaction (use!).

Kieren (1980) has postulated a structure for mathematical knowledge which is referred to as the five faces of mathematical knowledge building. The first face (mathematical face) exposes the student to various representations which may be physical, visual or symbolic in nature. The second face (psychological mechanism) manifests itself as an ability to process information using logical

structures and memory. The third face involves constructive mechanisms, and appears to be fundamental to increasing mathematical knowledge. (Kieren cites the counting behaviour or mechanism as an example of this face.) The fourth face is that of mental patterns or images. And the fifth face ties together the various informal language structures to those which are formal; this is the face of understanding grammars and languages. It is prima facie that GMATCH is capable of attending to the imagery face, but it would also appear that GMATCH could serve as a vehicle for addressing the remaining faces of mathematical growth.

In another work, Kieren (1983) has hypothesized that "there is a close relationship between the axioms of a mathematical system and the elements of intuition of the system expressed in terms of constructive mechanisms, images and hieroglyphic and analogical language use". Tentatively, it would appear that the use of GMATCH can foster intuitive notions of concepts and help to link these to more formal structures. It is postulated that the binding of the visual component with hand movements (recall that the graphics tablet is the preferred means of input at the current time) helps to stimulate cognitive development. Thus, the experiential domain of GMATCH can actively involve imagery, motion, coordination, and practice with entities or processes that are otherwise quite incorporeal (such as geometrical constructions).

An example from algebraic logic will serve to illustrate the potential for GMATCH. Logic is usually introduced at the senior high school level, with the formal notation typically supported by truth tables or Venn diagrams. Students quickly move from working with images to the manipulation of concise, symbolic expressions. With the exception of the elementary identities, students generally have difficulties in understanding and manipulating the formal notation, which leads one to assume that the fundamental links between the axioms and theorems with the mental images that are useful in dissecting the problem are not adequately formed. Since GMATCH has the potential to provide both static and animation (dynamic processes) displays, it would seem reasonable to assume that GMATCH can foster mental imagery using various Venn diagram displays, which are then actively connected to the formal symbolic structures. (This occurs by establishing what Bork refers to as a dialogue between student and computer system.) Furthermore, it can support both low-level and high-level symbolic manipulation (i.e. from simple hieroglyphs to complex geometric structures).

In conclusion, GMATCH can serve a dual role. Firstly, as a tool for extending current concepts of instruction by providing an image-oriented means for input and secondly, as a research tool for studying children's mechanisms of knowledge building. Thus, GMATCH might be viewed as a bootstrap device with the knowledge gained from the research

end supporting the improvement of instruction (supporting improvements to GMATCH itself). Yet, these are some of the potentials of GMATCH; in its current form, it can only be viewed as a forerunner for vastly improved input mechanisms that will extend the boundaries of educational research.

BIBLIOGRAPHY

Agui, T., Nakajima, M., & Arai, Y. (1982). An algebraic approach to the generation and description of binary pictures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4, 635-641.

Alt, F. L. (1962). Digital pattern recognition by moments. *Journal of the Association for Computing Machinery*, 9, 240-258.

Armstrong, J.M. (1980). *Achievement and participation of women in mathematics: An overview*. Education Commission of the States.

Barnea, D. I., & Silverman, H. F. (1972). A class of algorithms for fast digital image restoration. *IEEE Transactions on Computers*, 21, 179-186.

Blum, H. (1967). A transformation for extracting new descriptors of shape. In W. Wathen-Dunn (Ed.), *Models for the perception of speech and visual form* (pp. 362-380). Cambridge, Ma.: The MIT Press.

Bork, A. (1975). Learning through graphics. In R. J. Seidel & M. L. Rubin (Eds.), *Computers and communications: Implications for education* (pp. 287-301). New York: Academic Press.

Bork, A. (1981). Education technology and the future. *Journal of Educational Technology Systems*, 10, 3-20.

Bozinovic, R., & Srihari, S. N. (1982). A string correction algorithm for cursive script recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4, 655-663.

Brown, C. M. (1981). Some mathematical and representational aspects of solid modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3, 444-453.

Carpenter, T. P., & Osborne, A. R. (1975). Needed research on teaching and learning measure. In R. A. Lesh & D. A. Bradbard (Eds.), *Number and measurement: Papers from a research workshop* (pp. 85-100). Georgia Center for the Study of Learning and Teaching Mathematics: ERIC/SMEAC.

Cook, N., & Kersh, M. E. (1980). Improving teachers' ability to visualize mathematics. In R. Karplus (Ed.), *Proceedings of the fourth international conference for the psychology of mathematics education* (pp 377-383). University of Berkeley, Ca.

Davis, A. (1982). *Elf v4. documentation*. Edmonton: University of Alberta, Division of Educational Research Services.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: John Wiley & Sons.

Edmonds, E. A., Schappo, A., & Scrivener, A. R. (1982). Image handling in two-dimensional design. *IEEE Computer Graphics and Applications*, 2(5), 75-88.

Fennema, E., & SHerman, J. (1977). Sex-related differences in mathematics achievement, spatial visualization and socio-cultural factors. *Journal of Educational Research*, 14, 51-71.

Fennema, E., & Sherman, J. (1978). Sex-related differences in mathematics achievement and related factors: A further study. *Journal for Research in Mathematics Education*, 9, 189-203.

Foley, J. D., & Van Dam, A. (1982). *Fundamentals of interactive computer graphics*. Reading, Ma.: Addison-Wesley.

Fields, C., & Paris, J. (1981). Hardware - software. In H. F. O'Neil, Jr. (Ed.), *Computer-based instruction: A state-of-the-art assessment* (pp. 65-90). New York: Academic Press.

Freeman, H. (1974). Computer processing of line drawing images. *Computing Surveys*, 6, 57-97.

Fu, K.-S. (Ed.). (1982). *Applications of pattern recognition*. Boca Raton, Fl.: CRC Press.

Fu, K.-S., & Bhargava, B. K. (1973). Tree systems for syntactic pattern recognition. *IEEE Transactions on Computers*, 22, 1087-1099.

Gaines, B. R. (1977). Foundations of fuzzy reasoning. In M. M. Gupta, G. N. Saridis, & B. R. Gaines (Eds.), *Fuzzy automata and decision processes* (pp. 19-75). New York: Elsevier North-Holland.

Grosky, W. I., & Jain, R. (1983). Optimal quadtrees for image segments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5, 77-83.

Hall, E. L. (1979). *Computer image processing and recognition*. New York: Academic Press.

Huggins, W. H. (1974). What is needed? *Proceedings of the Battelle Computer Graphics Conference*, 8(1), 32-44. (Summary)

Hull, J. J., & Srihari, S. N. (1982). Experiments in text recognition with binary n-gram and Viterbi algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4, 520-530.

Hull, J. J., Srihari, S. N., & Choudhari, R. (1983). An integrated algorithm for text recognition: Comparison with a cascaded algorithm, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5, 384-395.

Jacobsen, J. D. (1968). Geometric relationships for retrieval of geographic information. *IBM Systems Journal*, 331-341.

Kashyap, R. L., & Oommen, B. J. (1982). A geometrical approach to polygonal dissimilarity and shape matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4, 649-654.

Kieren, T. E. (1981). *Five faces of mathematical knowledge building*. (Occasional Paper No. 20). Edmonton: University of Alberta, Department of Secondary Education.

Kieren, T. E. (1983). *Axioms and intuition in mathematical knowledge building*.

Kim, C. E. (1982). Digital convexity, straightness, and convex polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4, 618-626.

Kovalevsky, V. A. (1980). *Image pattern recognition* (A. Brown, Ed. and trans.). New York: Springer-Verlag.

Lee, D. (1983). *Draw2* [Computer program]. Edmonton, Alberta, Canada: Division of Educational Research Services, Faculty of Education, University of Alberta, Edmonton. (Report No. RIR 83-1)

Lee, E. T. (1974). The shape-oriented dissimilarity of polygons and its application to the classification of chromosome images. *Pattern Recognition*, 6, 47-60.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics - Doklady*, 10, 707-710.

Loy, W. W., & Landau, I. D. (1982). An on-line procedure for recognition of handprinted alphanumeric characters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4, 422-427.

Mallgren, W. R. (1982). *Formal specification of interactive graphics programming languages*. Cambridge, Ma.: The MIT Press.

Markowsky, G., & Wesley, M. A. (1980). Fleshing out wire frames. *IBM Journal of Research and Development*, 24, 582-597.

Micro Control Systems, Inc. (1982). MCS space tablet (advertisement). *Byte*, 7(12), 412-413.

Nevins, A. J. (1979). An orientation free study of handprinted characters. *Pattern Recognition*, 11, 155-164.

Nevins, A. J. (1982). Region extraction from complex shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4, 500-511.

Nygard, K. E., & Ranganathan, B. (1983). A system for generating instructional computer graphics. *AEDS Journal*, 16, 177-187.

Olson, A. T. (1981). The geometry of shape. *International Journal of Mathematics Education and Science Technology*, 12, 355-368.

Pavlidis, T. (1968). Analysis of set patterns. *Pattern Recognition*, 1, 165-178.

Requicha, A. A. G., & Voelcker, H. B. (1982). Solid modeling: A historical summary and contemporary assessment. *IEEE Computer Graphics and Applications*, 2(2), 9-24.

Romaniuk, E. W. (1970). *A versatile authoring language for teachers*. Unpublished doctoral dissertation, University of Alberta, Edmonton.

Rosenfeld, A. (Ed.). (1976). *Digital Picture Analysis*. Berlin: Springer-Verlag.

Sampson, J. R. (1976). *Adaptive information processing: An introductory survey*. New York: Springer-Verlag.

Schonberger, A.K. (1976). The interrelationship of sex, visual spatial abilities and mathematical problem solving ability in grade seven. *Dissertation Abstracts International*, 37, 3536A.

Simon, J. C. (1975). Recent progress to formal approach of pattern recognition and scene analysis. *Pattern Recognition*, 7, 117-124.

Skemp, R. R. (1975). *The psychology of learning mathematics.* Middlesex, England: Penguin.

Sleeman, D., & Brown, J. S. (1982). *Intelligent tutoring systems.* New York: Academic Press.

Sugarman, R. (1978). A second chance for computer-aided instruction. *IEEE Spectrum*, 15(8), 29-37.

Sugihara, K. (1982). Mathematical structures of line drawings of polyhedrons--toward man-machine communication by means of line drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4, 458-469.

Tanaka, H., Hirakawa, Y., & Kaneku, S. (1982). Recognition of distorted patterns using the Viterbi algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1982, 4, 18-25.

Thomason, M. G. (1973). Finite fuzzy automata, regular fuzzy languages, and pattern recognition. *Pattern Recognition*, 5, 383-390.

Tsai, R. Y. (1983). Multiframe image point matching and 3-d surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5, 159-174.

Udupa, K. J., & Murthy, I. S. N. (1975). Some new concepts for encoding line patterns. *Pattern Recognition*, 7, 225-233.

Uhr, L., & Vossler, C. (1963). A pattern-recognition program that generates, evaluates, and adjusts its own operators. In E. A. Feigenbaum, & J. Feldman (Eds.), *Computers and thought* (pp. 251-268). New York: McGraw-Hill.

Voyce, S. (1980). A multilingual-interpreter system for languages used in computer assisted instruction. (Doctoral dissertation, University of Toronto, 1979). *Dissertation Abstracts International*, 40, 4311A.

Wakayama, T. (1982). A core-line tracing algorithm based on maximal square moving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4, 68-74.

Wallace, T. P., Mitchell, O. R., & Fukunaga, K. (1981). Three-dimensional shape analysis using local shape descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3, 310-323.

WICAT Inc. (1982). *Wise Notes*.

Yau, M.-M., & Srihari, S. N. (1983). A hierarchical data structure for multidimensional digital images. *Communications of the ACM*, 26, 504-515.

Appendix A: Glossary

artificial intelligence: Marvin Minsky's definition is "the science of making machines do things that would require intelligence if done by men".

ASCII: An acronym for American Standard Code for Information Interchange. It is based upon an eight bit code.

author: The term given to a programmer who constructs CAI course lessons for the purpose of delivering instruction to a student.

authoring: The process, attributed to an author, of creating CAI lessons on a computer system.

bit: The smallest unit of memory. A single "cell" whose state is either on (1) or off (0). A binary digit.

bit block transfer: A transfer of data from one segment of memory to another segment of memory, which occurs without regard to the logical byte boundaries.

bitplane: A designated segment of memory that is mapped to the video display. Writing data to this area causes the display to change.

byte: A grouping of eight contiguous bits of memory. This is often the fundamental block of memory.

compilation: The process of using a compiler program to convert a high-level language, source program (a program which is relatively machine-independent and not directly executable by a CPU) to a low-level object program (a highly machine-dependent program consisting of the binary codes which are decipherable and executable by the CPU).

courseware: This is a generic term referring to computer programs that are designed to provide instruction to students using a CAI system.

CPU: An abbreviation for the central processing unit, which is essentially the heart of a computer system. It is responsible for controlling the interpretation and execution of computer code (programs).

CPU time: The actual time required for the central processing unit to perform some task. This is not the same as real-time (i.e. clock time) since a CPU may spend much of its time servicing other processes

(time-sharing), thus the real-time may be quite long while the CPU time is short.

CRT: An abbreviation for cathode ray tube, that is usually used in reference to the television-like device attached to a terminal (also monitor).

computer-assisted instruction (CAI): A mode of teaching or learning controlled and directed by a computer system using interactive terminals to present student lessons and test acquisition of concepts or skills. The basic CAI styles include drill and practice, tutorial and simulation. Also referred to as computer-based instruction (CBI), computer-aided learning or computer-assisted learning (CAL).

digitizing tablet (also graphics tablet): A flat, rectangular, pad-like device to which is connected either a pen cursor (reminiscent of a ball point pen) or a hand-held puck. Moving the pen cursor across the surface of the tablet causes the tablet to generate Cartesian coordinate data indicating the current location of the pen. Programs written to function with a graphics tablet are capable of creating complex graphics images.

DMA (Direct memory access): The transfer of data from memory directly to peripheral devices without CPU intervention (the CPU is only required to initially set up the conditions for transfer); also known as cycle-stealing.

graphics tablet: (see digitizing tablet)

grammar: The basic set of rules which govern the generation of languages. A grammar is formally defined by a quadruple (V,T,P,S) which represent the set of variables, terminal symbols, productions (rules for writing), and the start symbol, respectively.

inflection point: The point on a curve at which a tangent line separates the curve into two regions; one which is concave upward and the other which is concave downward.

lightpen: An electro-optical device resembling a wired ballpoint pen that is capable of "reading" pixels of light on a CRT and rendering this information into x, y coordinates.

metric: A relation d on a set of points (x,y) is a metric if the following conditions are met:

1. $d(x,y) = 0$ iff $x = y$.
2. $d(x,y) = d(y,x)$.
3. $d(x,y) + d(y,z) \Rightarrow d(x,z)$

mouse: A generic term applied to any of the hand-held, puck-like devices that are designed to replace the conventional keyboard means of input. Such devices may be either electromechanical or electro-optical in nature -- similar in function to a lightpen.

multi-tasking: A description applied to a computer system capable of performing more than one task or process simultaneously per user.

operating system: The underlying software or programs that provide a computer system with its functionality.

optical character recognition: A process of recognizing printed type (sometimes, script) based upon optical scanning devices and accompanying hardware to filter out noise and to align the images.

pattern recognition: No single outstanding formal definition is agreed upon by the practitioners in this area. As far as this study is concerned, the closest definition is "the process which groups some patterns or stimuli into certain categories according to some intrinsic properties which they possess".

RAM: An abbreviation for random access memory. The primary memory of a computer is RAM; data may be stored (written to) or retrieved (read) from RAM. Data stored in RAM is not permanent, and is usually lost as soon as the power is shut off.

real-time: The time normally measured by humans (i.e. clock-time based on the 24 hour day).

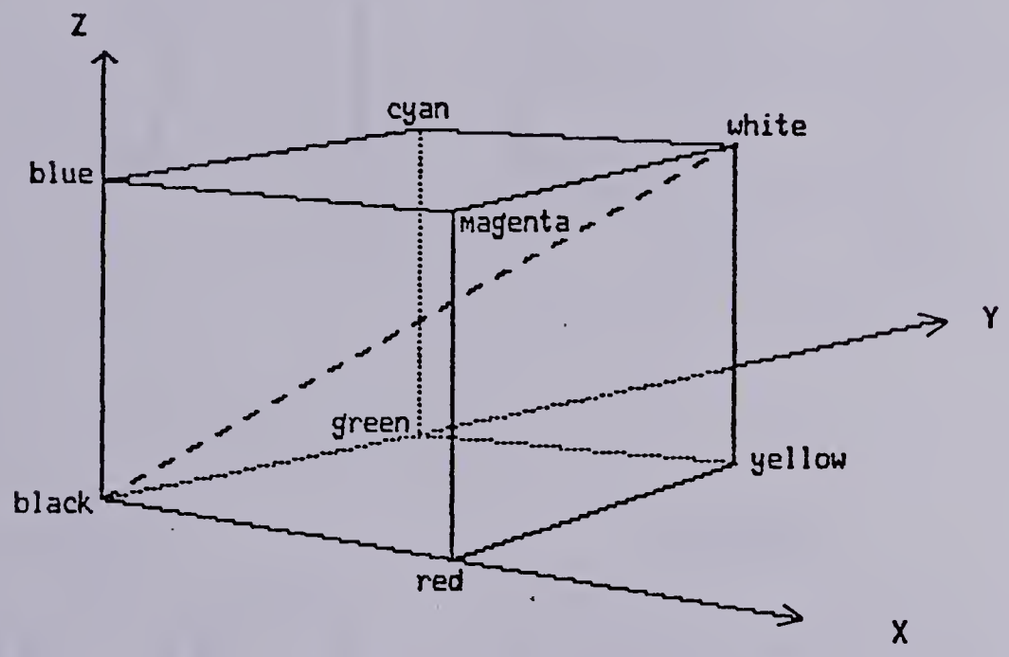
software: Initially used as a generic term for computer code (programs) as opposed to the physical devices of a computer system. Now the term is often used in reference to the specific programs which provide a computer system with its operational power, that is, the operating system, the file editors, etc.

touch-sensitive screen: An analog of the lightpen which may be based upon a fine grid of wires placed on top of the CRT surface. Pressing on this grid with your fingers generates x, y coordinate data to the computer system or terminal.

wireframe: The wireframe of a three-dimensional object is its set of edges and vertices.

Appendix B: The RGB color space

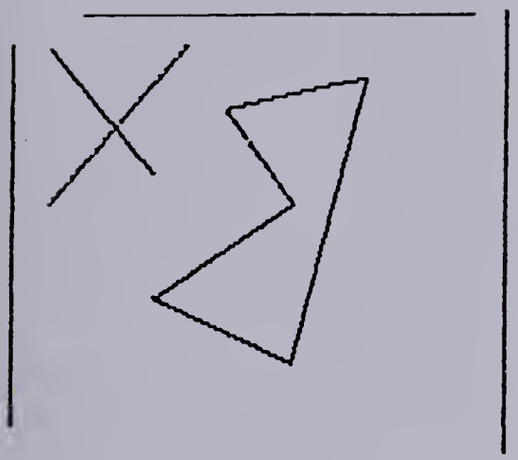
The RGB color model is based on a three-dimensional Cartesian coordinate system using a unit cube with the primaries (red, green, and blue) lying along the x, y, and z axes respectively as shown below (adapted from Foley & Van Dam, p. 611).



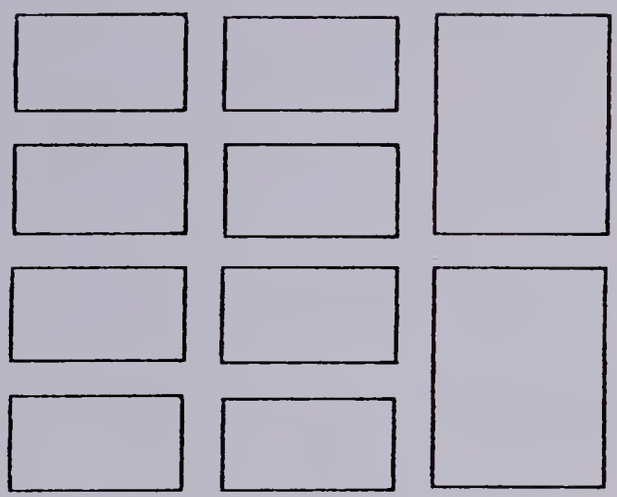
The distance along each axis indicates the proportion of color contribution from the respective primary. The various grey levels are represented by the main diagonal which receives equal contributions from each primary. This color space is additive in nature, that is, the contributions of each primary are added to form the resultant color (e.g. equal amounts of red and green produce shades of yellow).

Appendix C: The 10-Primitives Target Sets

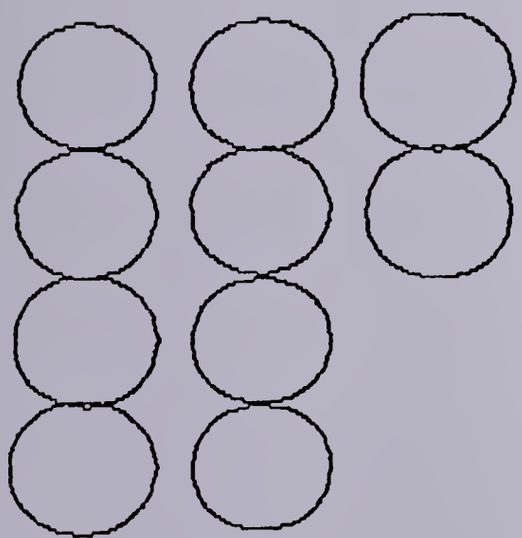
VECTORS



RECTANGLES



CIRCLES



COMBINATION